Driving Value

IEC 61131-3 Basics with MotionWorks IEC

Class Number: TRM011-MotionWorksIEC-Basic
Doc ID: eLV.MotionWorksIEC.01.Basic
Rev 1.10
Date: March 6, 2014

YASKAWA

 © 2011 Yaskawa Electric America, Inc. All rights reserved.

Contents

- *Everything but motion*
  - *In this class we won't have servos, axes, or make anything move*
  - *You'll manipulate bits, values, and time*
  - *You'll become proficient in the programming and navigation of MotionWorks IEC*
  - *You'll experience the IEC 61131-3 languages, data types, and programming structure*
  - *When you see the IEC logo, this means the page is directly related to IEC 61131-3*

Driving Value

Software Orientation

Mpiec Controllers vs PLC simulator
Mwiec versions, install, registration tips
Mwiec main screen, simple program editing

YASKAWA

Install MotionWorks IEC

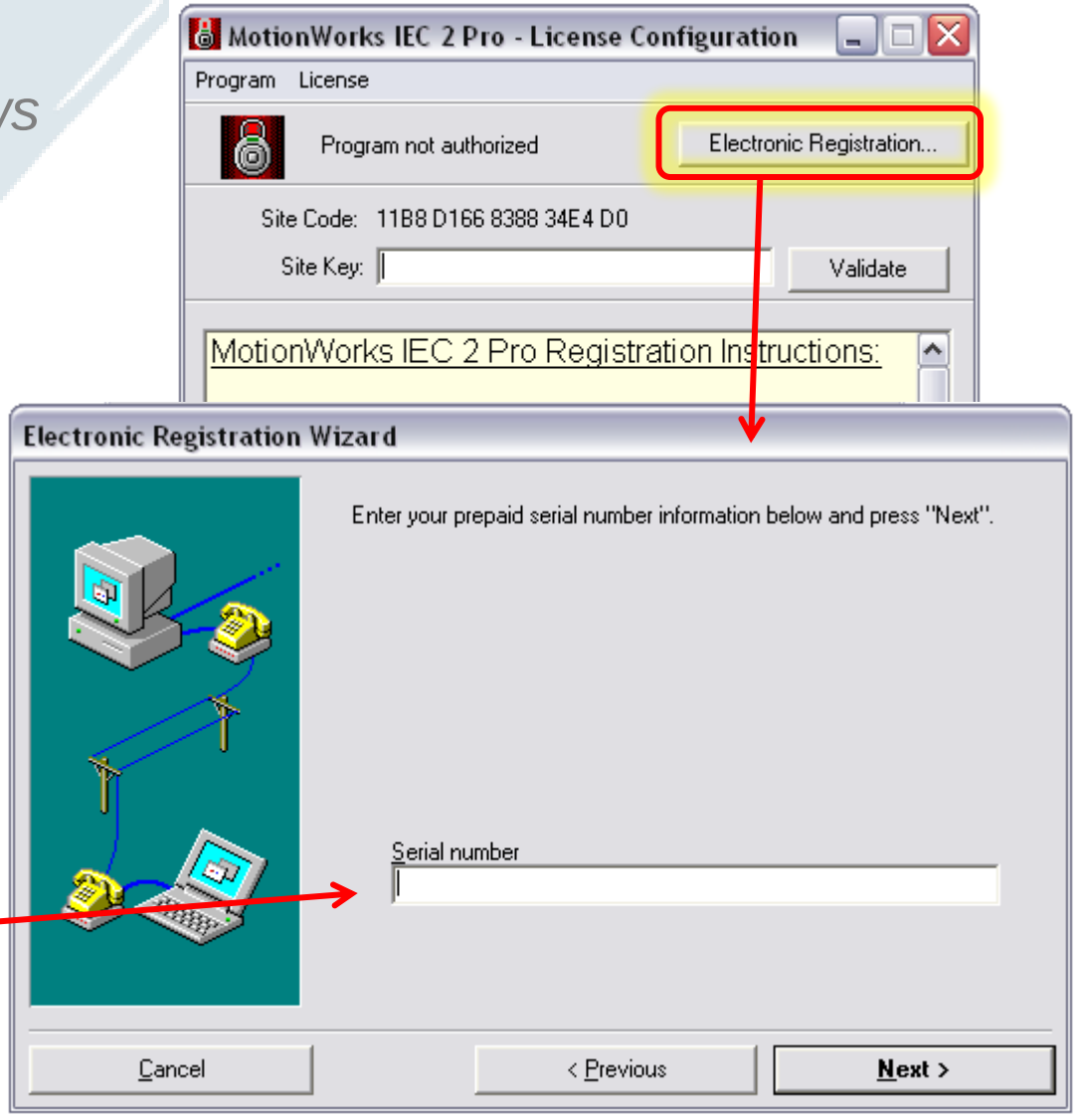
- *Download page*
  - *yaskawa.com/iecsw*
  - *Installation instructions*
- *Windows XP Requirements*
  - *Service Pack 3*
  - *Dot Net 3.5*
- *Windows 7*
  - *Multi-Language Support*

MotionWorks IEC Registration

- *30-day Demo*
  - *Do not change windows clock*
- *Unlimited License*
  - *$$ Cost*
  - *Serial Number*
  - *Internet Connection*
  - *USB license option*

Sales Order Email contains Serial Number

See Video: http://youtu.be/yGC2TmtP1bU?t=22s



Express & Pro

- *MotionWorks IEC "Express"*
  - *Simplified Interface*
  - *1 Execution Task*
  - *Fundamental programming languages (ST, LD)*
  - *Easy to use*
  - *Easy to learn*
  - *Low cost*
  - *For simple applications requiring point-to-point, camming, gearing, ethernet*

- *MotionWorks IEC "Pro"*
  - *Full options interface*
  - *Multi-tasking operation possible*
  - *All five IEC 61131-3 languages available (ST, LD, FBD, IL, SFC)*
  - *Full featured*
  - *Medium cost*
  - *For the most demanding and high-performance applications*

Express & Pro

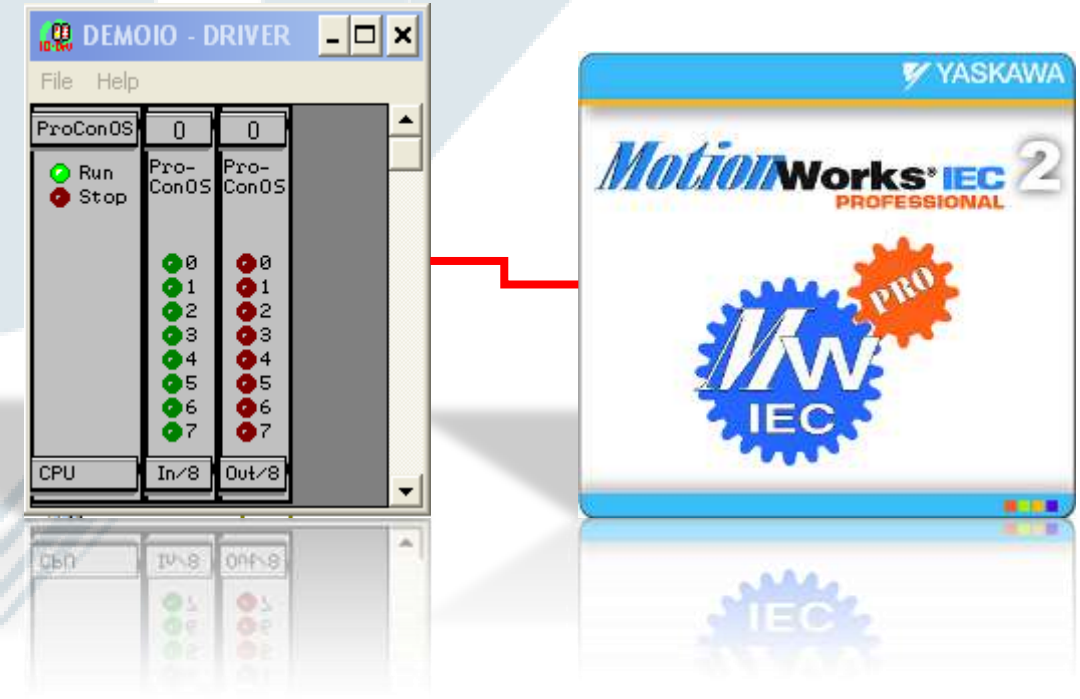
- *Both Express and Pro can be simultaneously installed in one PC*
  - *Separate license required for each*
- *Express projects can be opened in Pro*
  - *Express Project is converted to Pro format*
    - » *Save project under a new name or as zip project first*
  - *Pro projects cannot be opened with Express*
- *Code can be copied and pasted between Pro and Express*
  - *Multiple Express / Pro open at the same time*
  - *Easiest to copy the entire POU*

Ethernet Communication via PC

Web Server
- Confirm Ethernet communication
- Determine/Set IP address
- Troubleshoot and clear alarms
- Save/Restore Project Archive
- Test Motion
- Upgrade Firmware
- Requires Java

MotionWorks IEC
- Monitor and Edit Application Program
- Configure axes, IO, Network
- Troubleshoot and clear alarms
- Save/Restore Project Archive
- Test Motion

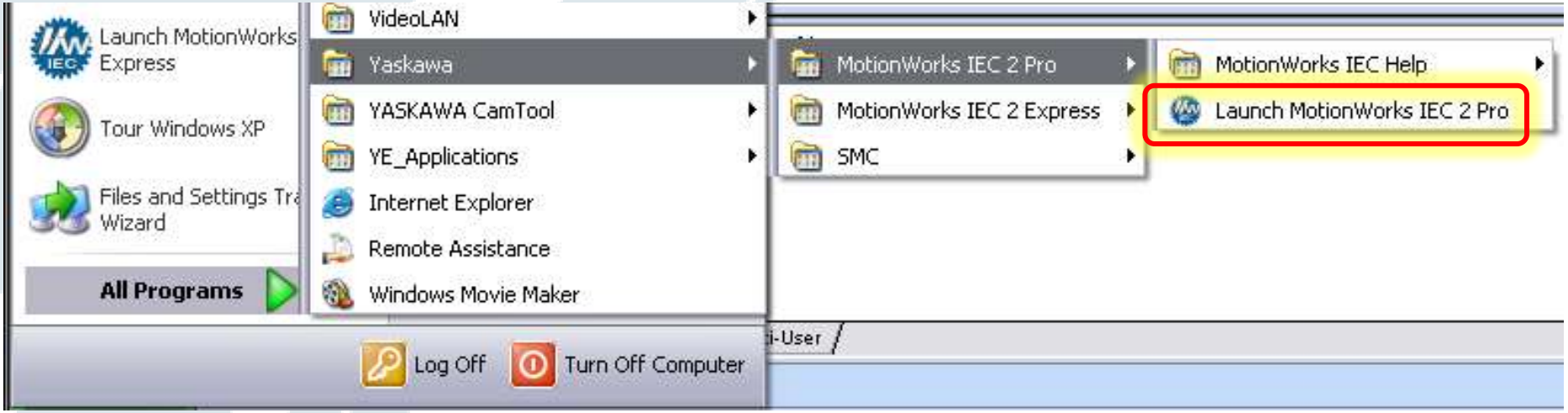
PLC Simulator



MotionWorks IEC

- Monitor and Edit Application Program
  - LD, ST, FBD, SFC languages
  - Tasks, Programs, Functions
  - I/O

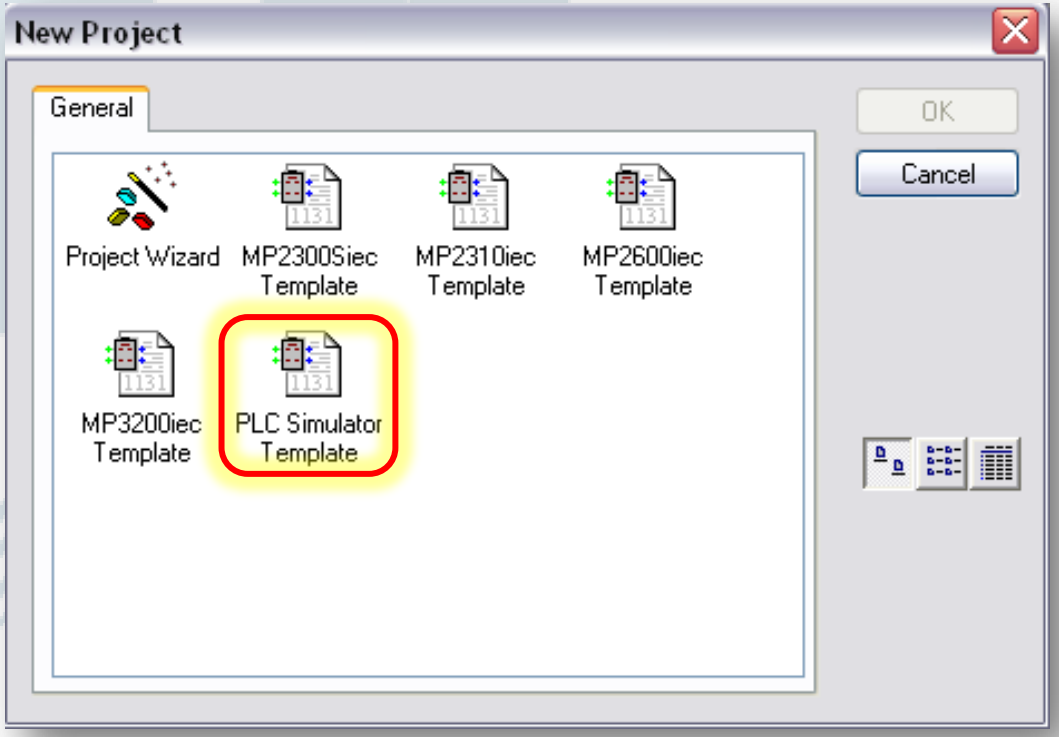
Start MotionWorks IEC

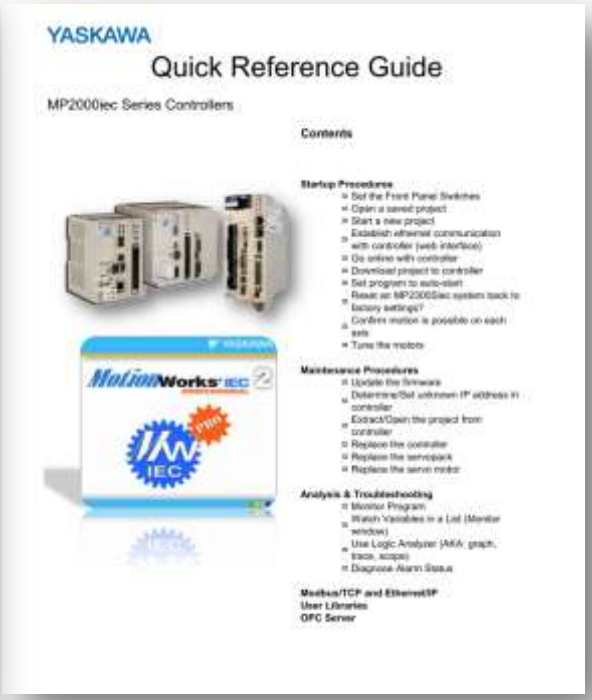


New Project

*Start a **new project***

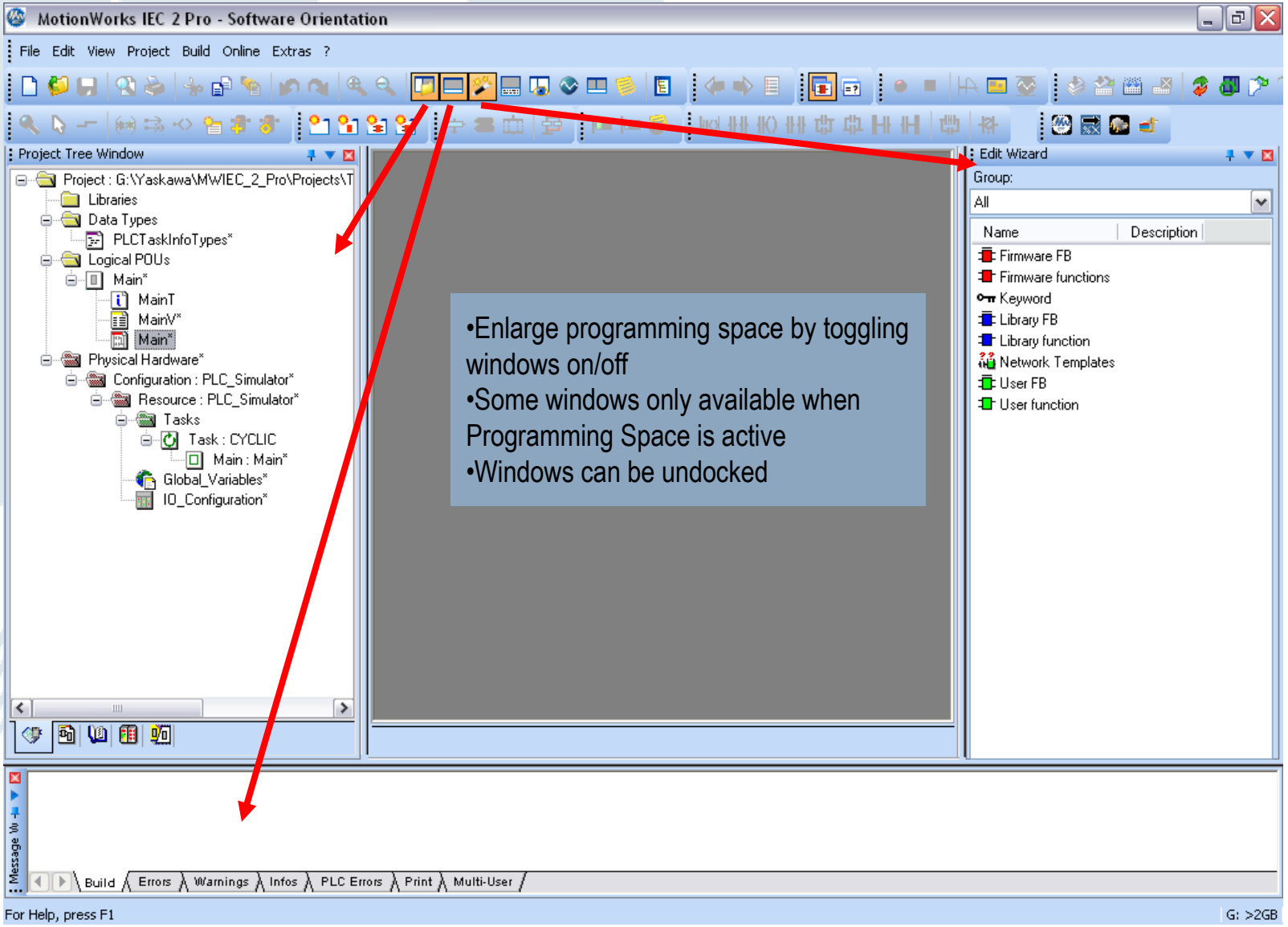
- *PLC Simulator Template*
- *File - Save Project As*



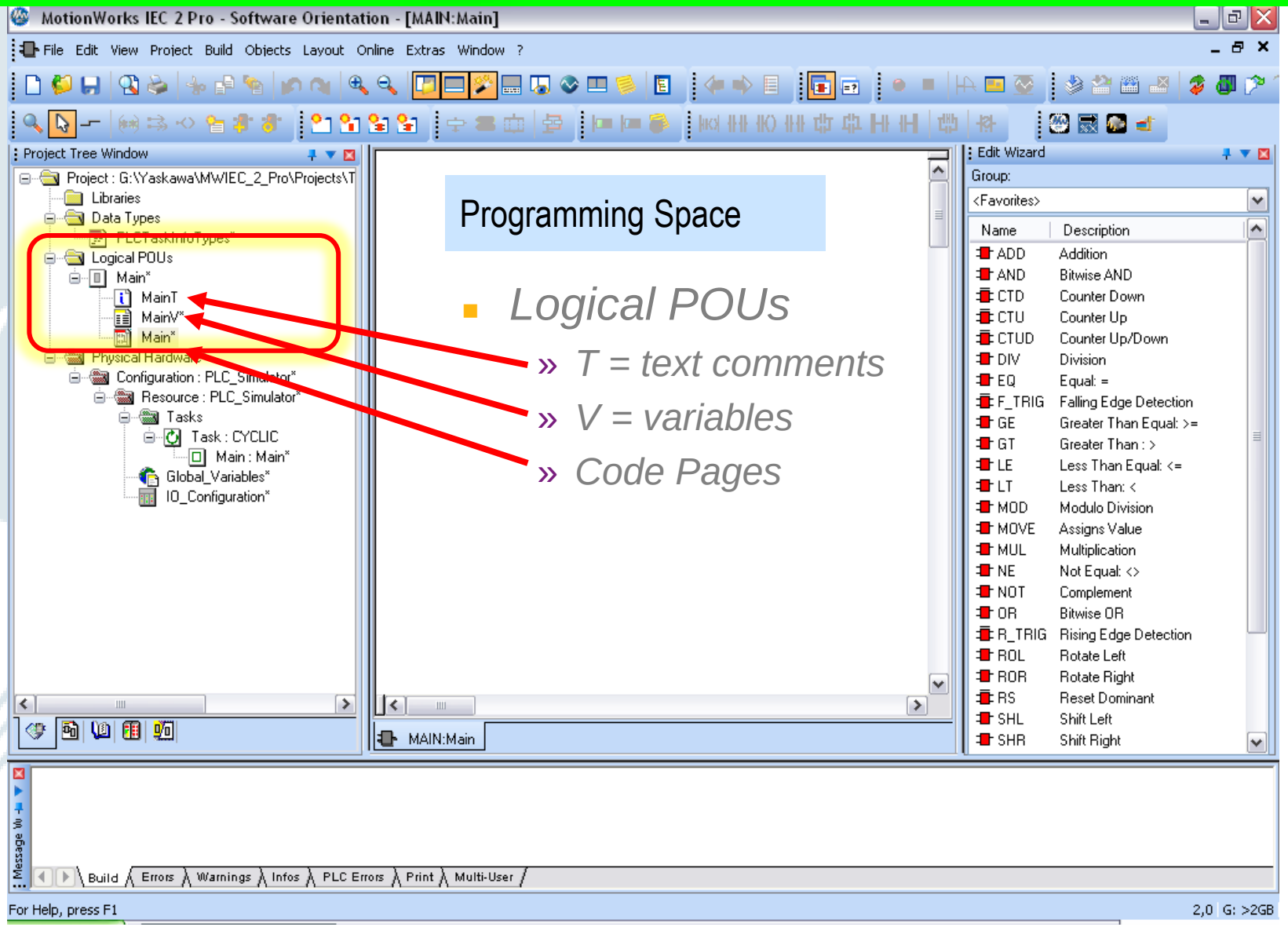


Software Orientation

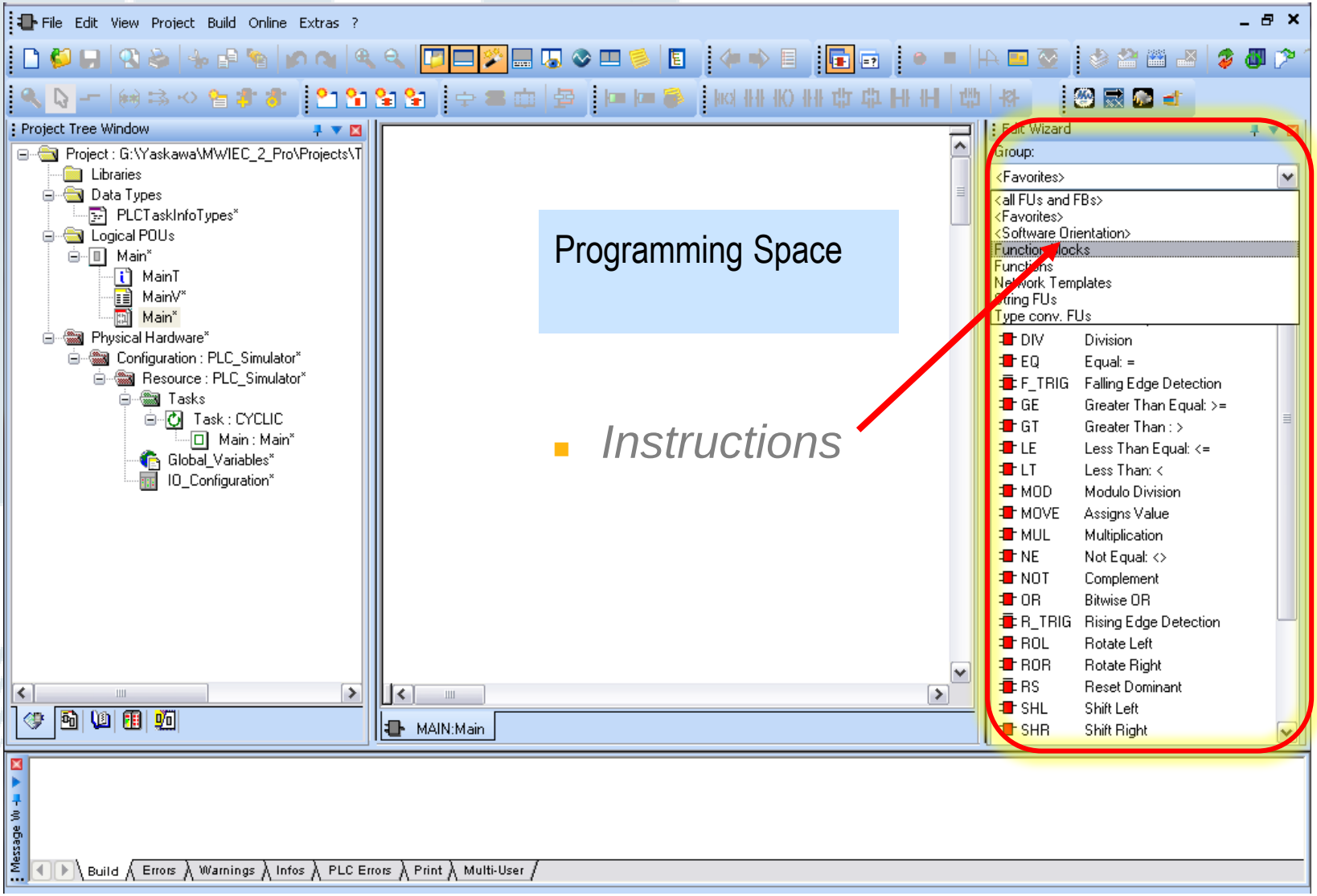
MotionWorks IEC Main Screen



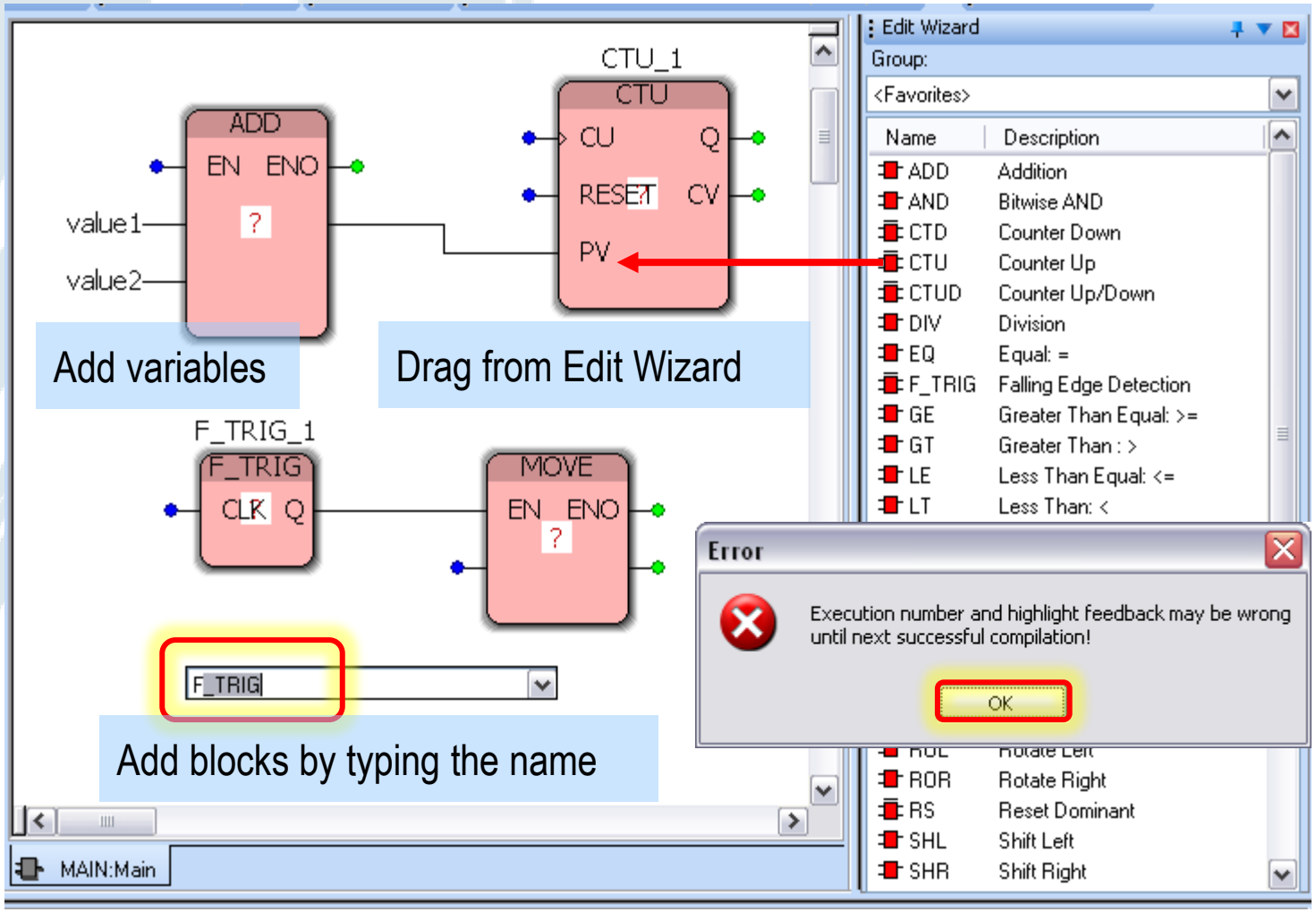
Software Orientation IEC MotionWorks IEC Main Screen



MotionWorks IEC Main Screen



Basic Ladder/Block Editing



Driving Value

Programming Quick Start

Function Block

Make, Download

Debug Mode

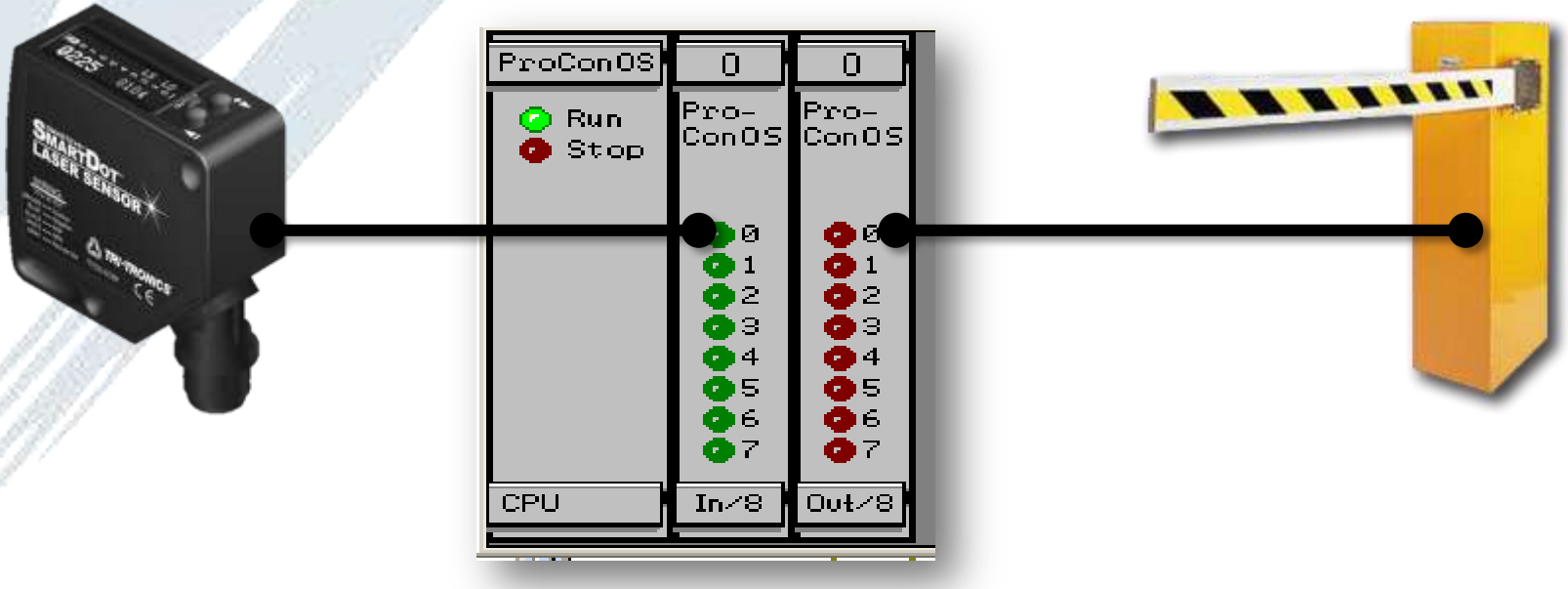
Ladder

Variables and Properties

YASKAWA

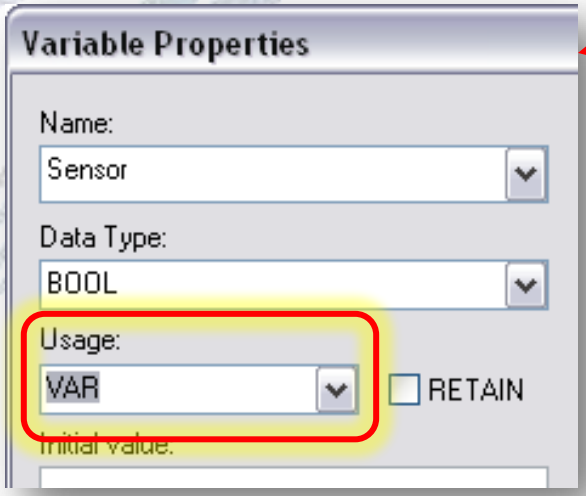
Overview

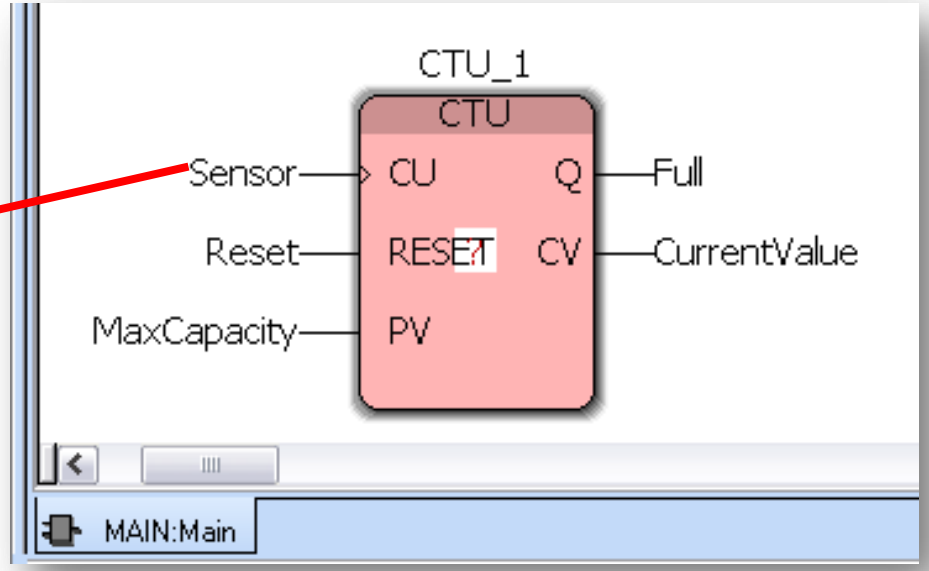
- *Learn the software by means of a simple program*
  - *Parking gate control*
  - *Many details to follow*



Counter Program

- *New Project – PLC Simulator Template*
- *POU "Main"*
  - *Counter UP block*
  - *Add variables (Usage: VAR)*
  - *Make, download, warm start*
  - *Debug Mode*
    - » *Enter values*



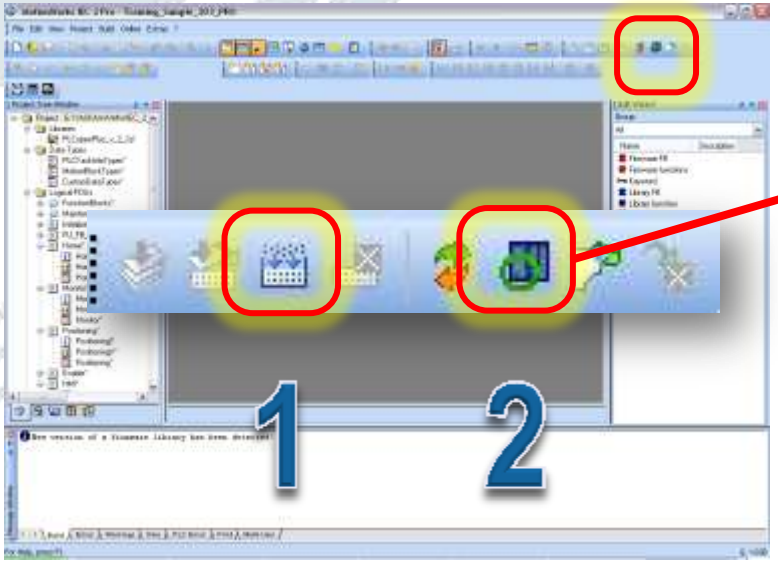


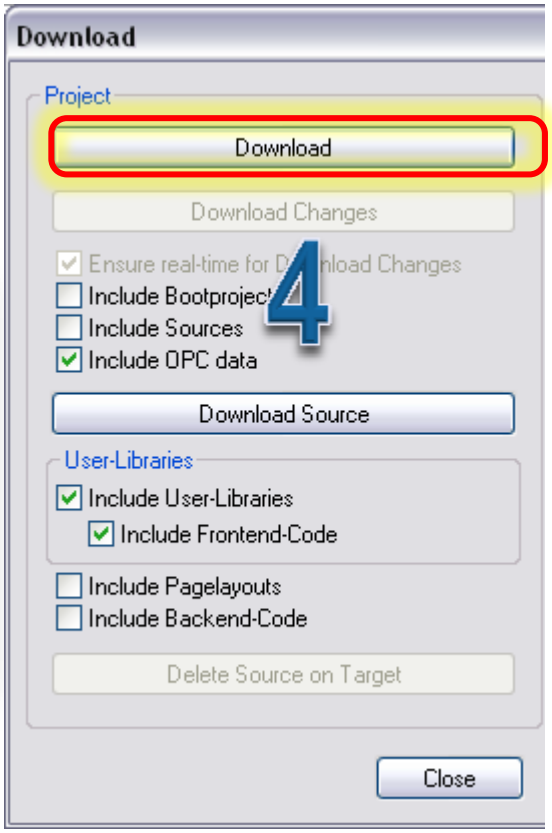
Add this code to the Main POU

Download

- *Steps to Download*
  1. *Make (includes automatic "save")*
  2. *Project Control ➔ Resource*
  3. *Download*
  4. *Download*

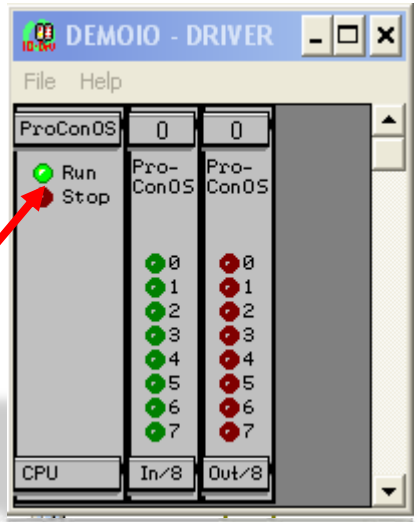
More details to come





Run PLC

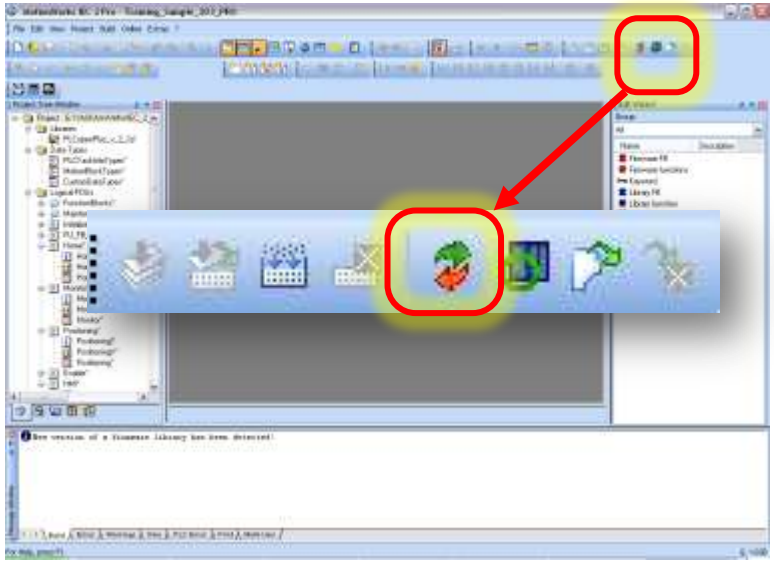
- *Warm Start*
  - *Normal “Power On” start*

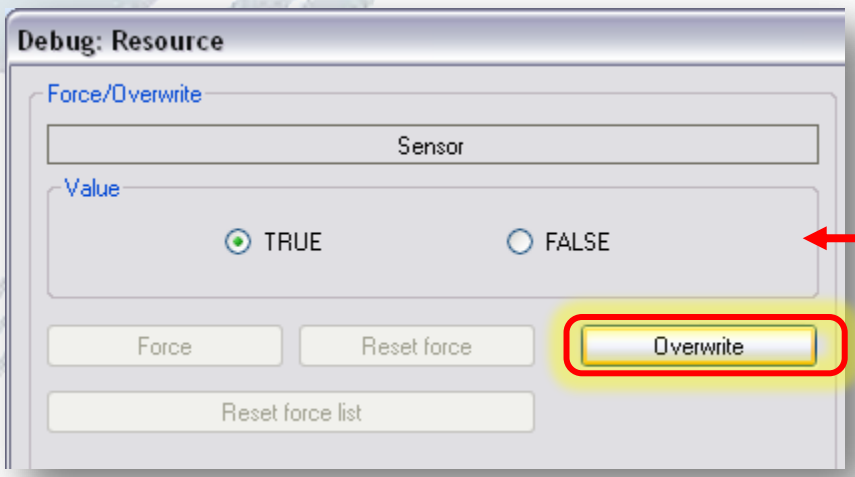


More details to come

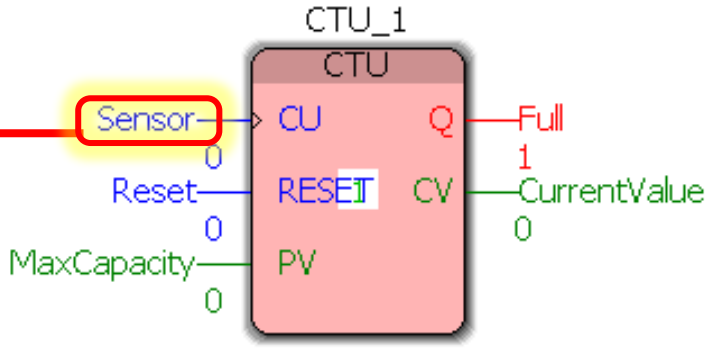
Monitor “Debug Mode”

- *Online – Debug*
- *Double click variables*
- *Overwrite*
- *Editing is disabled during debug*





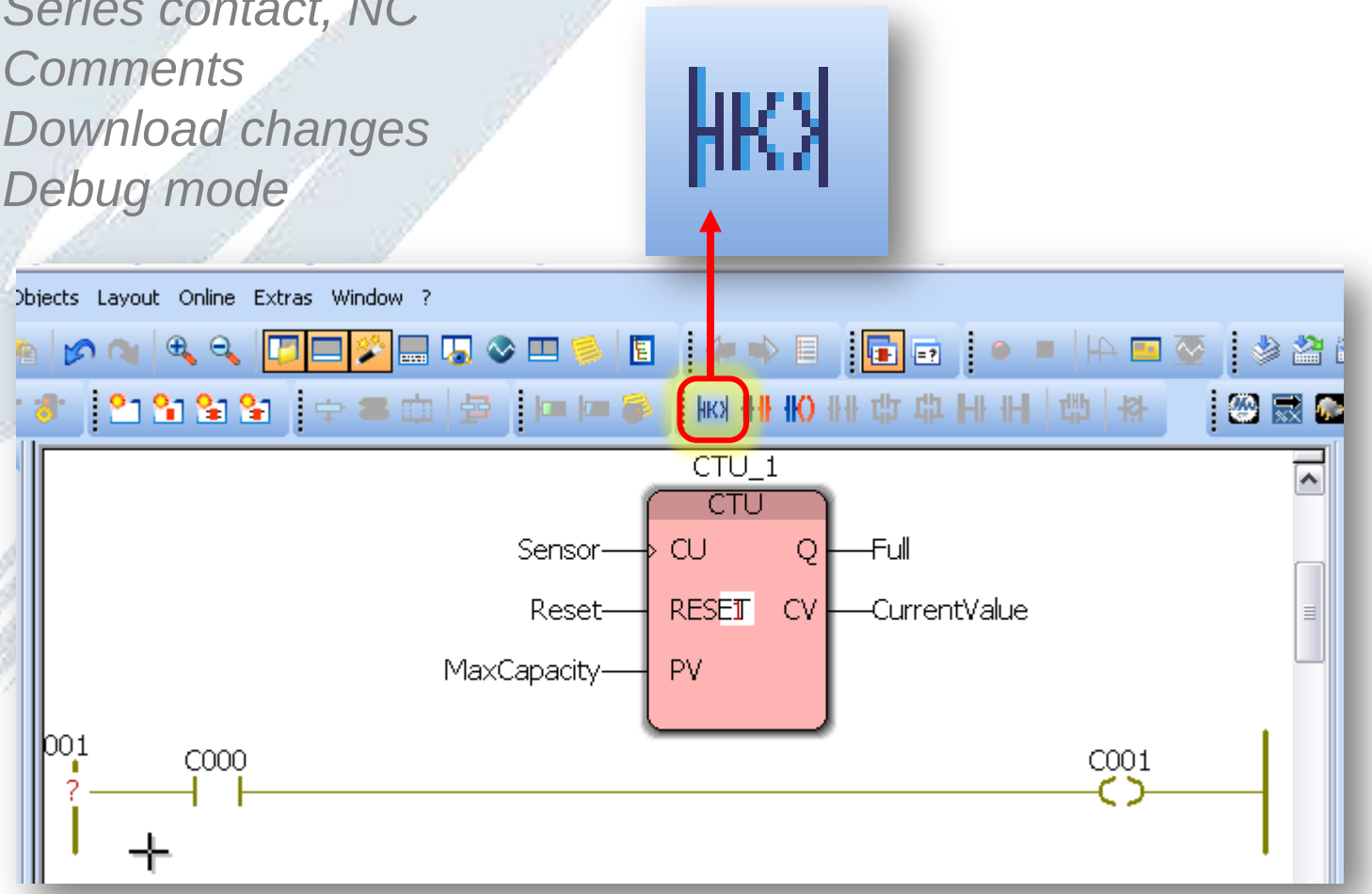
Simple “Parking Gate” Control



Debug Mode (Disables Editing)

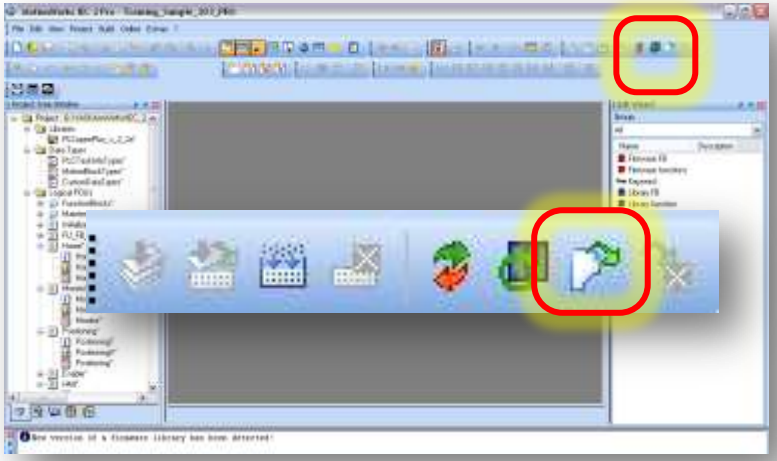
Simple Ladder

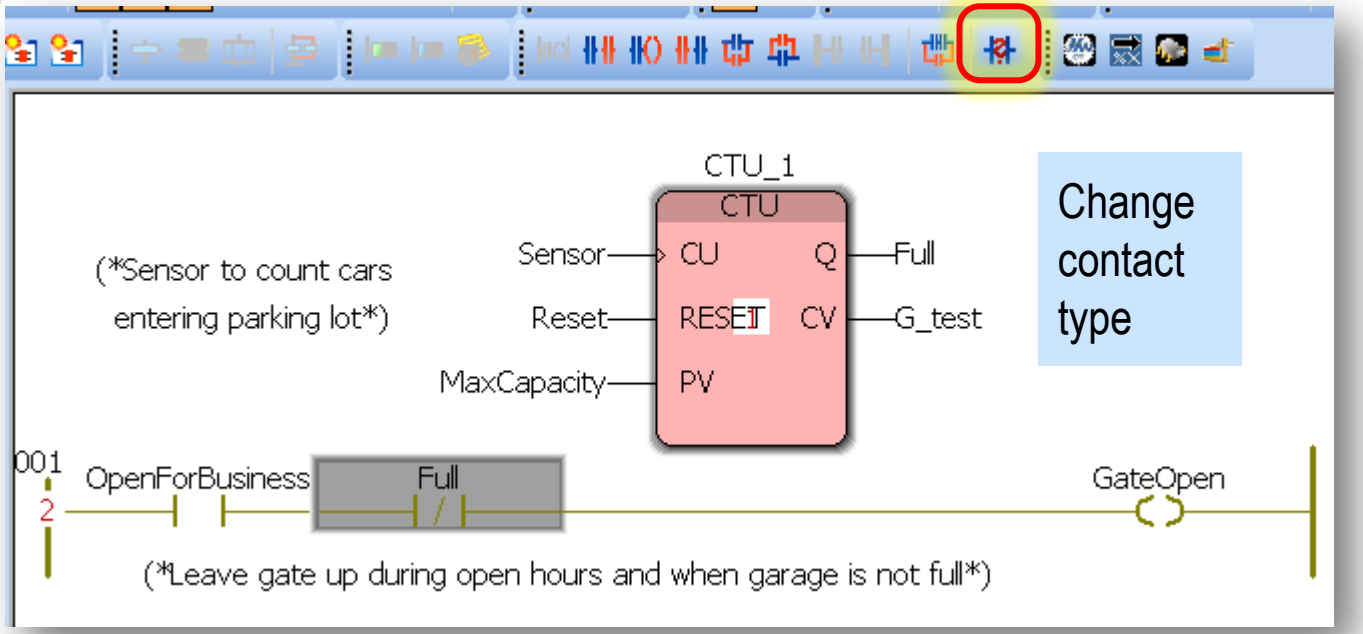
- *Simple Ladder*
  - *Network*
  - *Series contact, NC*
  - *Comments*
  - *Download changes*
  - *Debug mode*



Download Changes

- *Download Changes*
  - *Saves time*
  - *Includes*
    - » *Save*
    - » *Make*
    - » *Download*
  - *Does not require warm start*





Right-Click to add comment

Change contact type

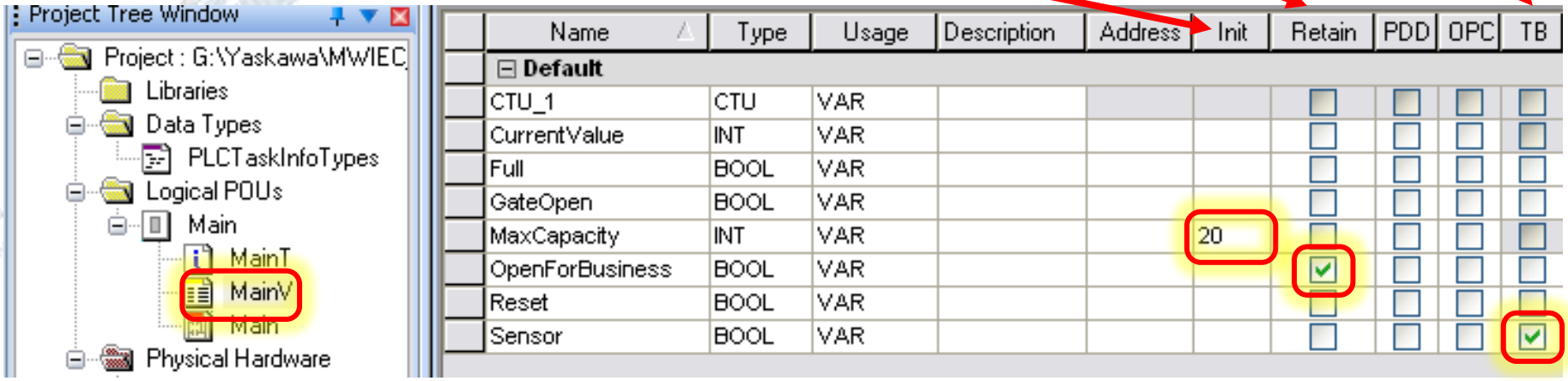
Local Variable Properties

- *Variable Properties*
  - *Toggle Boolean Mode*
  - *Retain*
  - *Initial Value*

Initial Value

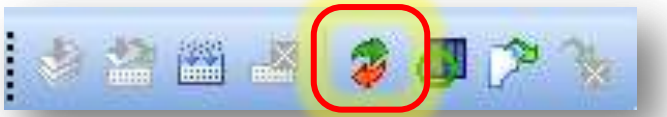
Retain last value after warm start

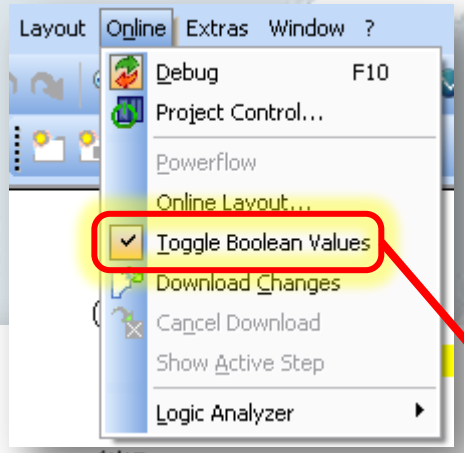
"Toggle Boolean" is a feature of debug mode



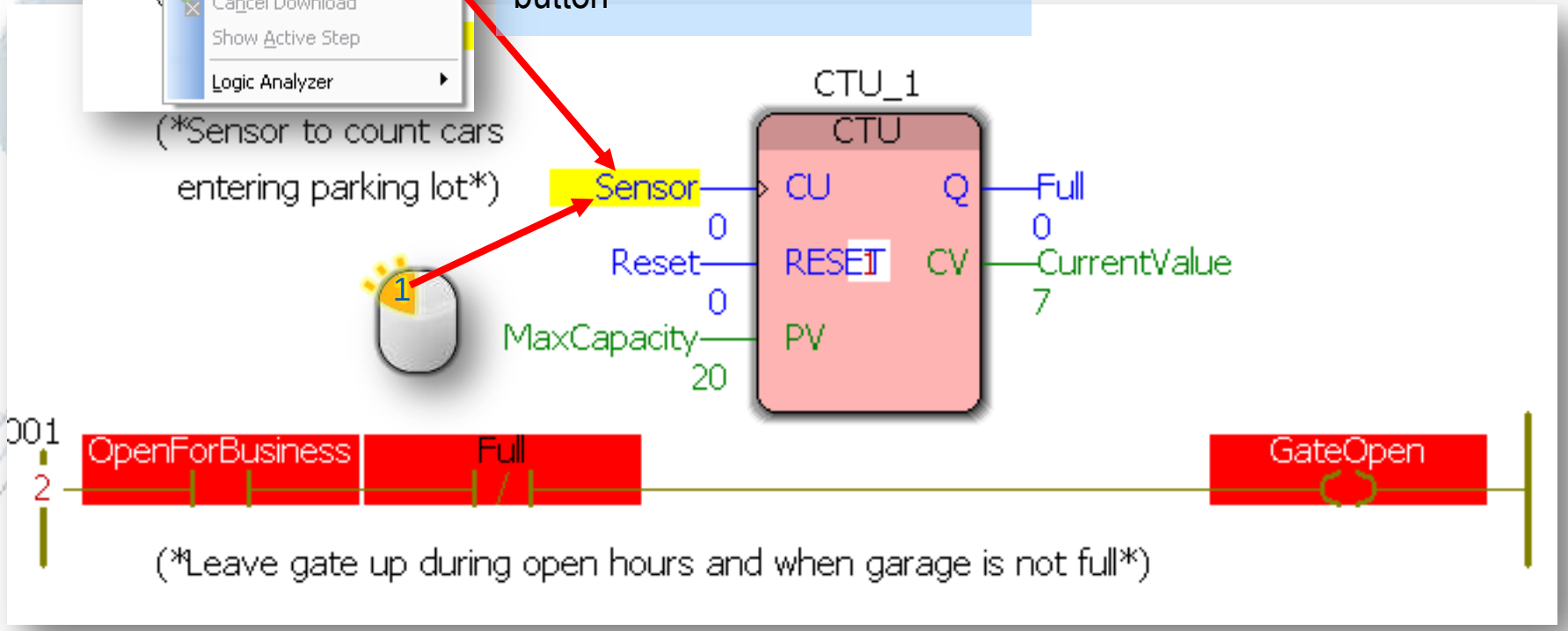
Toggle Boolean

- *Toggle Boolean*





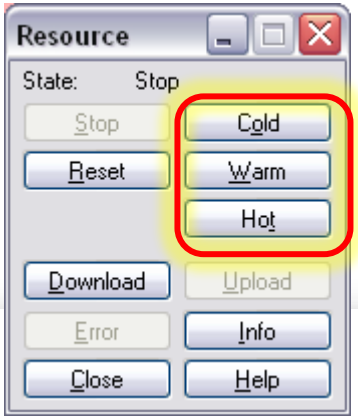
Toggle Boolean – variable in debug mode is like a push-button

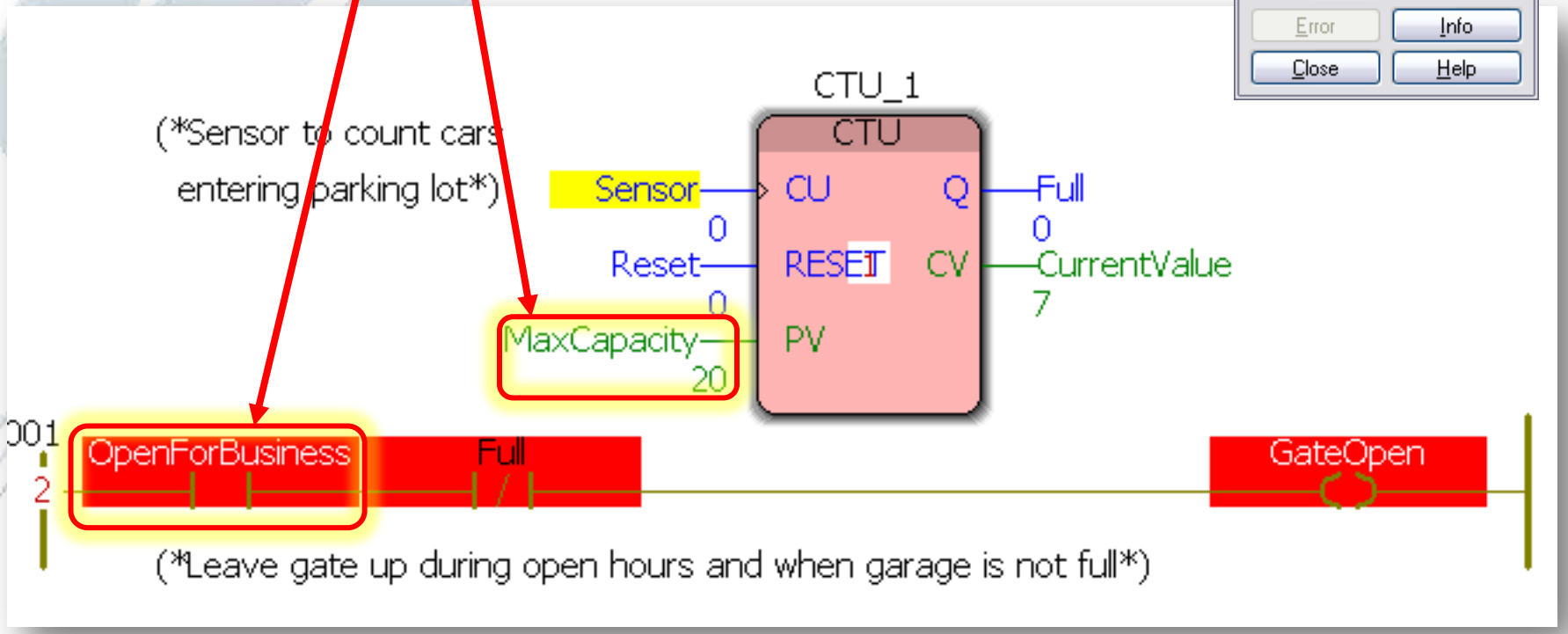


Variable Properties

- *Initial Value*

Change the value in debug mode.
What happens after Warm start?
Cold start?





Driving Value

Programming Quick Start #2

Task, Task Instance

Structured Text

F5 variable declaration

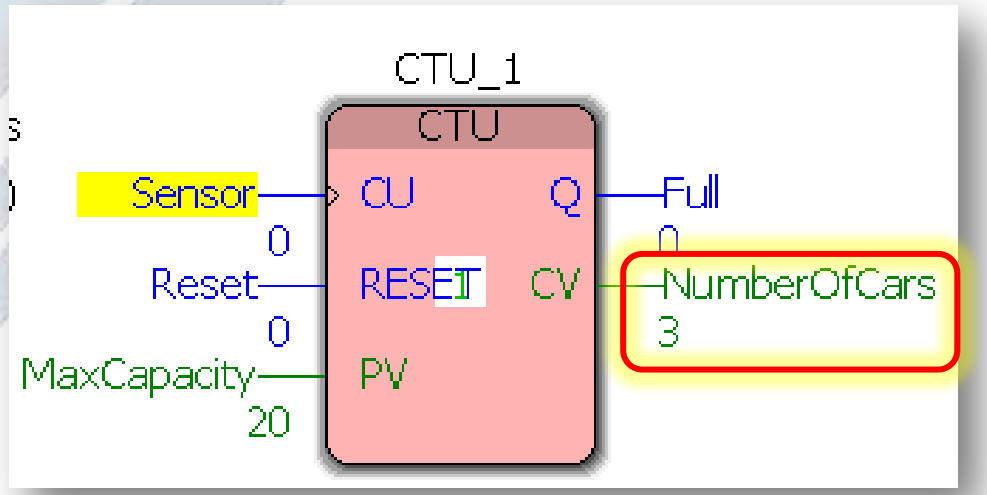
VAR & VAR_GLOBAL

Data Type & Conversion

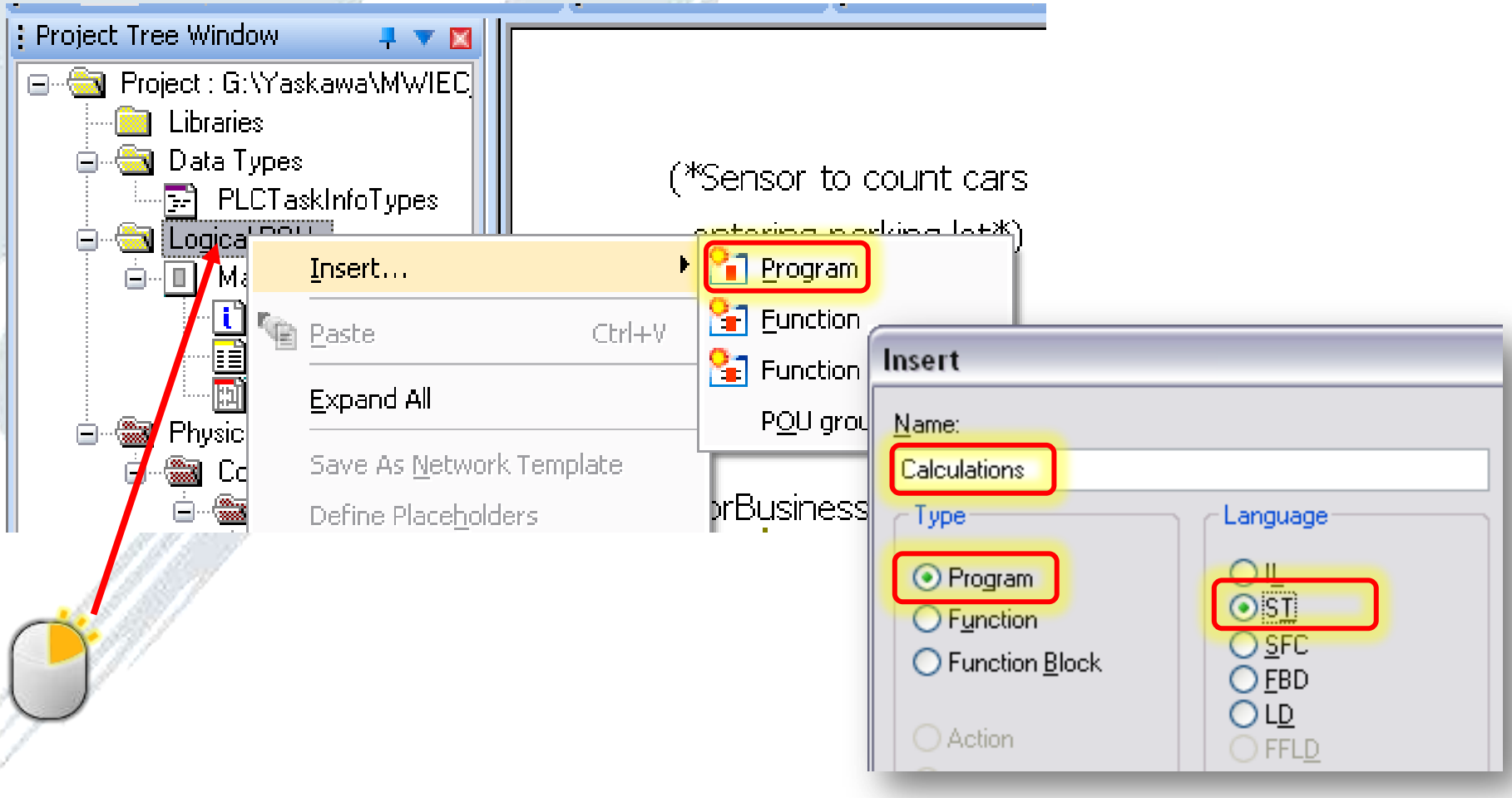
YASKAWA

New Application Challenge

- *Calculations in a Structured Text program (ST POU)*
  - *$29.99 per car*
    - » *Income = 29.99 * NumberOfCars*
  - *Ladder or Function block language also works*

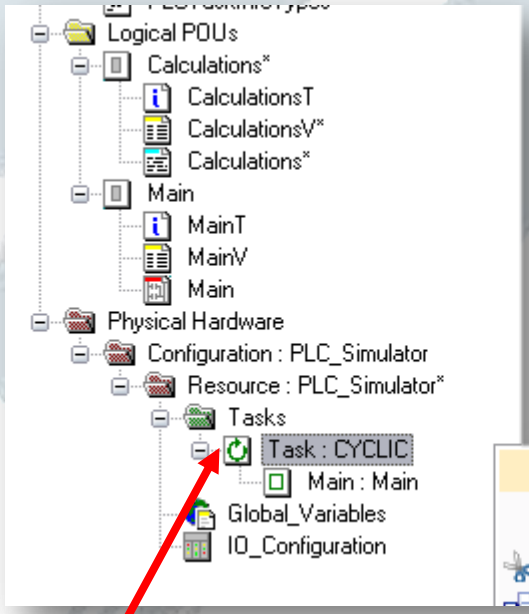


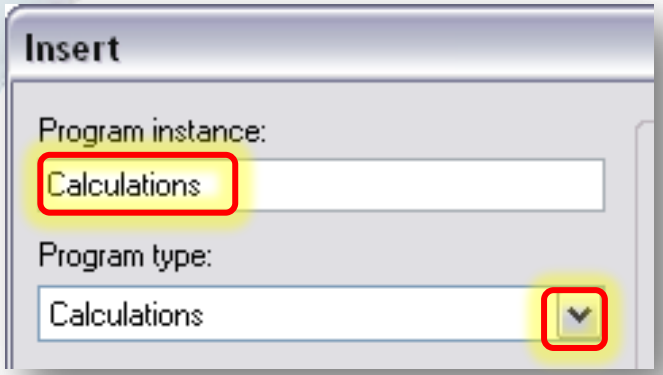
New ST Program POU



Program Instance

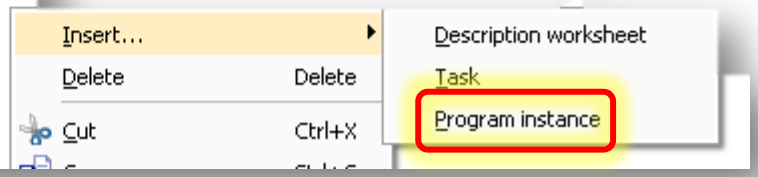
- *Program POU runs in a Task*
  - *Manually add program instance*





Enter “Program instance”

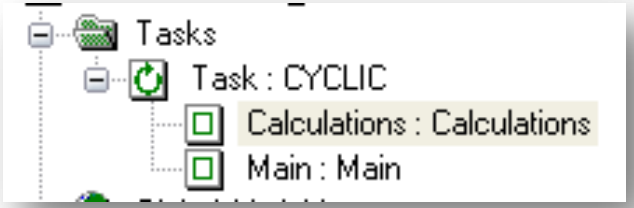
Choose “Program Type”



“Calculations” is not running in the task.

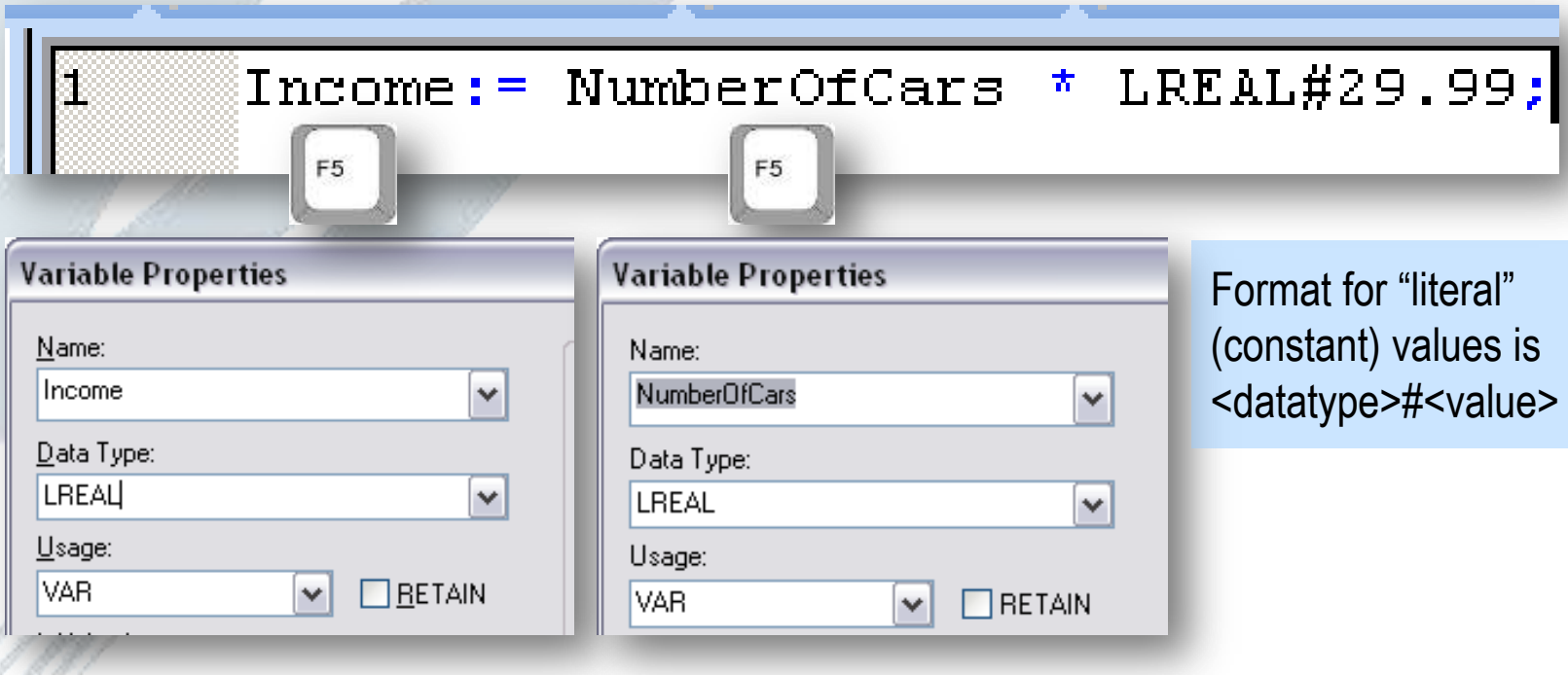
Insert a program instance.

Result



ST Code

- *Use F5 to declare variables*

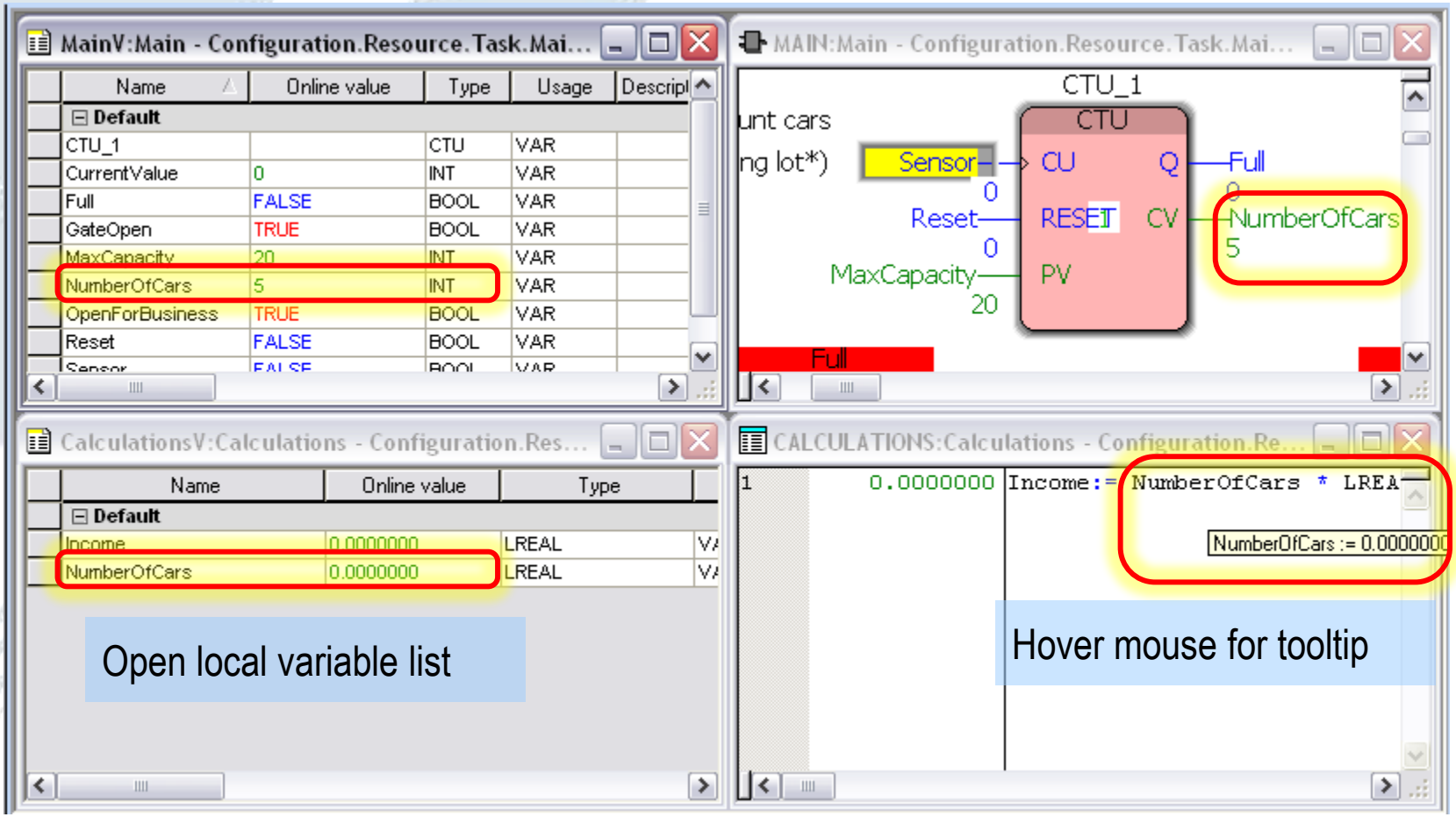


Format for “literal” (constant) values is <datatype>#<value>

Download Changes

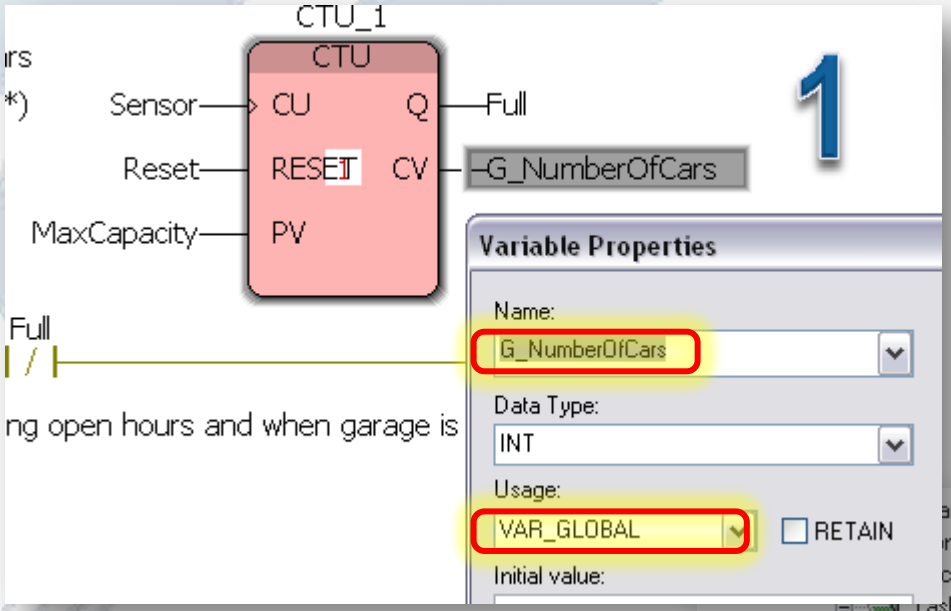
Local Variable

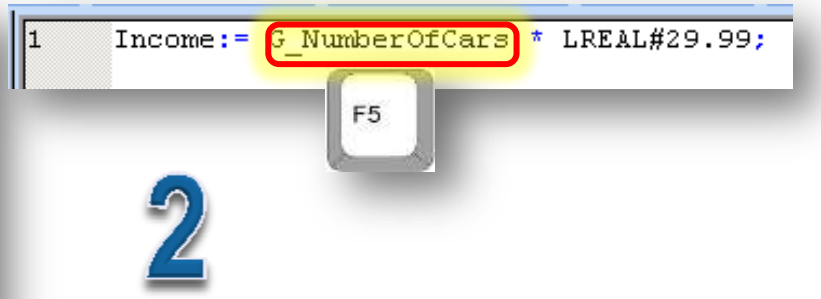
- *Why is "NumberOfCars" not the same value in both POUs?*

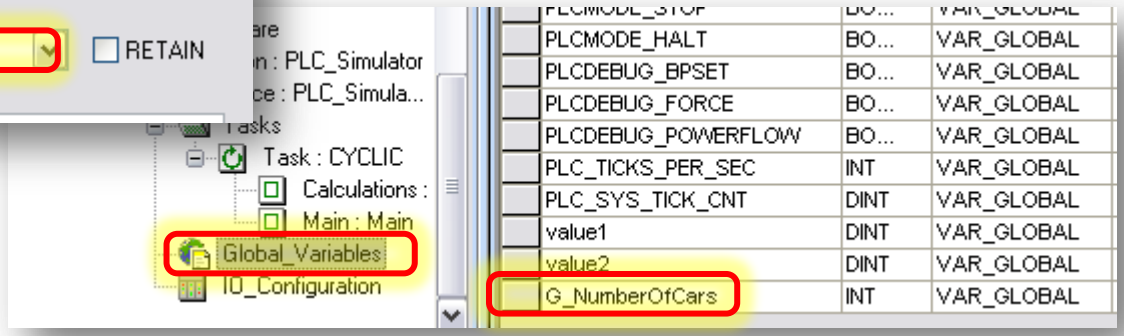


Global Variable

- *Use New Global Variable in both POUs*
  - *G_NumberOfCars*
  - *Global Variable List*





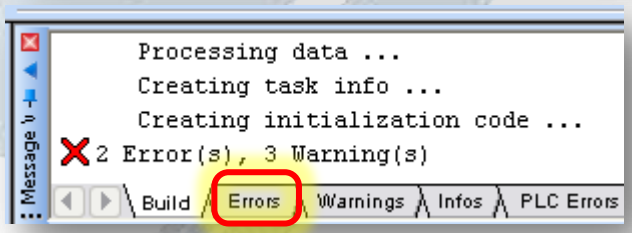


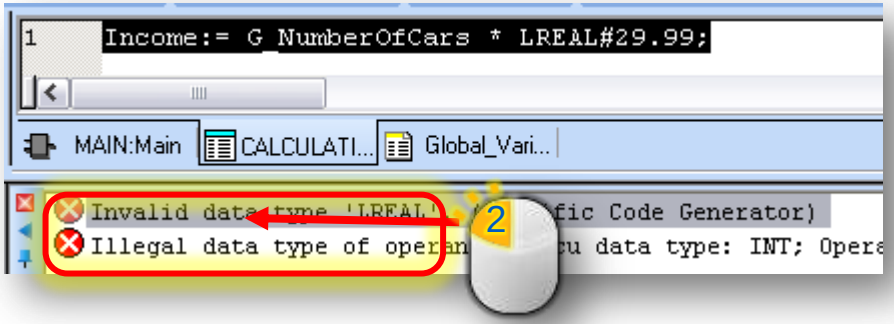
Best Practice: **G_** prefix

Data Type Invalid

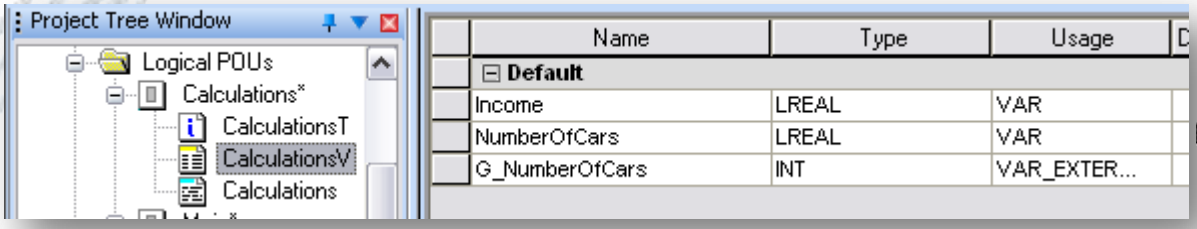
- *IEC 61131-3 "Strict Data Typing"*
  - *Operations on data with same data type only*
  - *Convert data type in code; no automatic conversion*

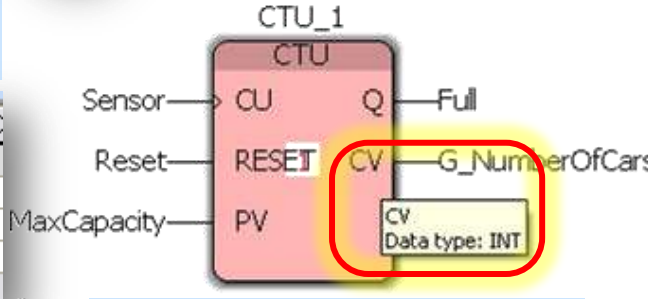
Download Changes





INT cannot multiply by LREAL

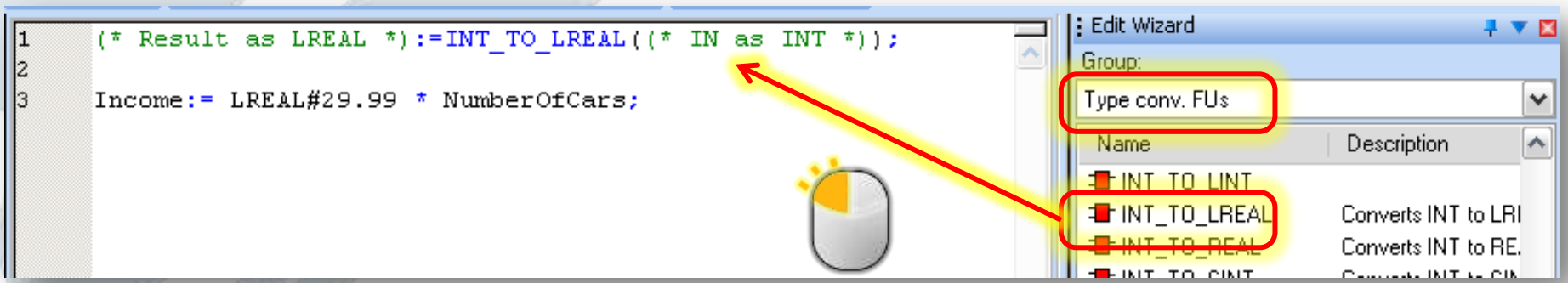


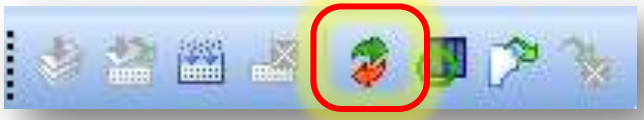


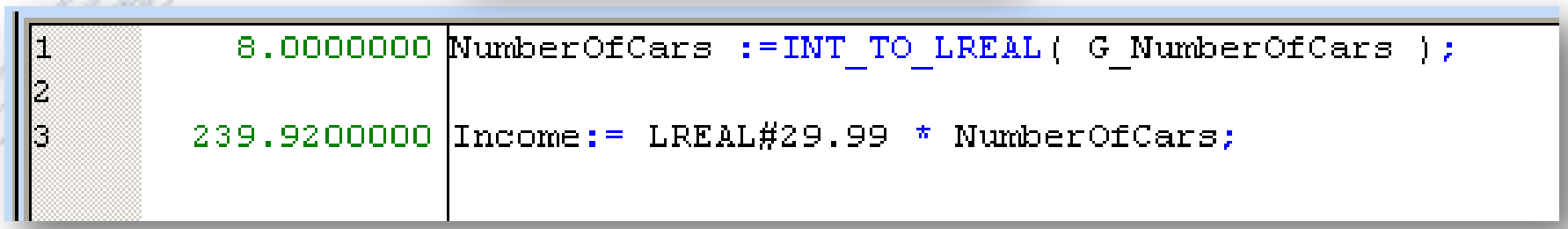
CV output datatype is INT

Data Type Conversion

- *Use Conversion Functions in code*
  - *Best Practice required by IEC 61131-3*







Driving Value

HELP DOCUMENTATION

Programming System

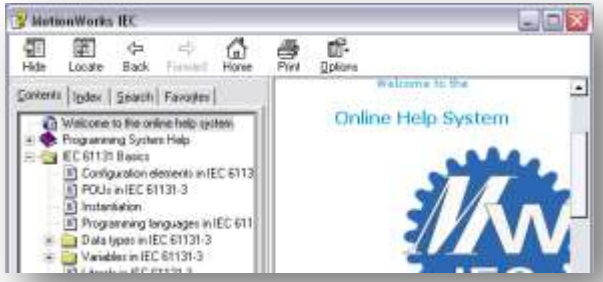
IEC 61131 blocks

Right-Click Help

Contextual Help Button

YASKAWA

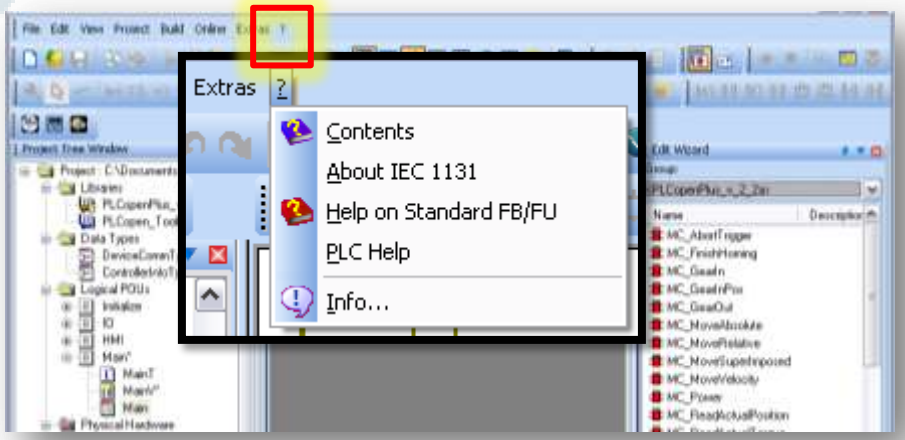
- *CHM Documentation Installed*
  - *MotionWorks IEC*
    - » *MPReadMe001.chm*
  - *Help on Functions and Function Blocks*
    - » *PLC_FBfun001.chm*
  - *Search for *.CHM*
    - » *C:\Program Files (x86)\Yaskawa*

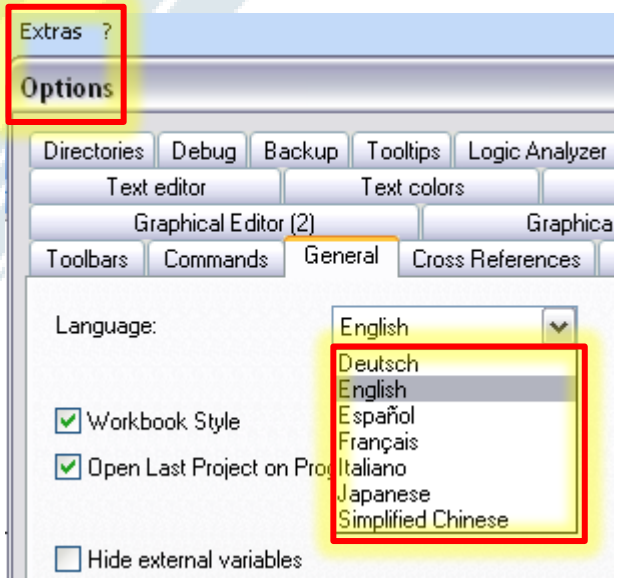


001 English
081 Japanese
086 Chinese

MotionWorks IEC Software

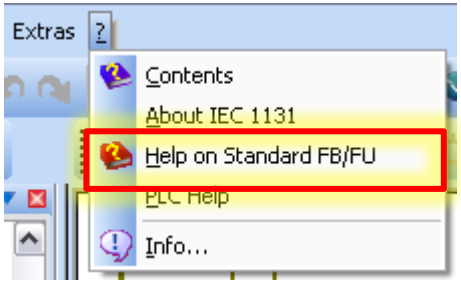
- *Programming System*
- *IEC 61131-3 Basics*
  - *“About IEC 1131”*
- *General PLC*
  - *“PLC Help”*
- *Available in several languages*
  - *Extras – Options – General - Language*

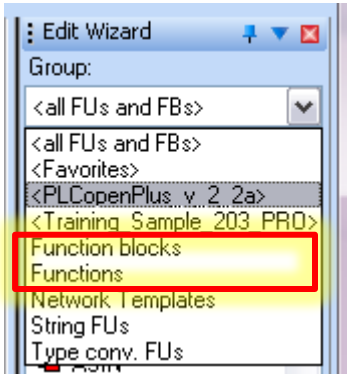


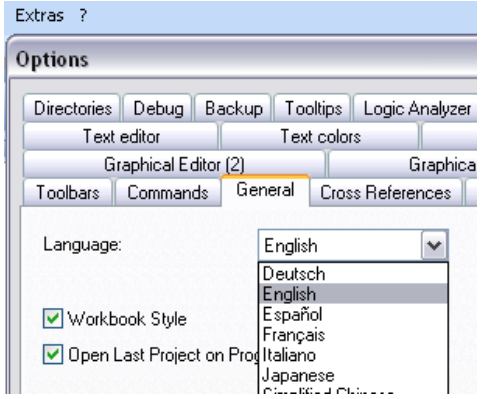


Functions and Function Blocks

- *Standard IEC 61131-3 blocks*
  - *Add, subtract, timer, pulse, …*
- *Applies to These Groups in Edit Wizard*
  - *Function Blocks*
  - *Functions*
  - *String FUs*
  - *Type conv. FUs*
- *Available in Several Languages*
  - *Extras – Options – General - Language*



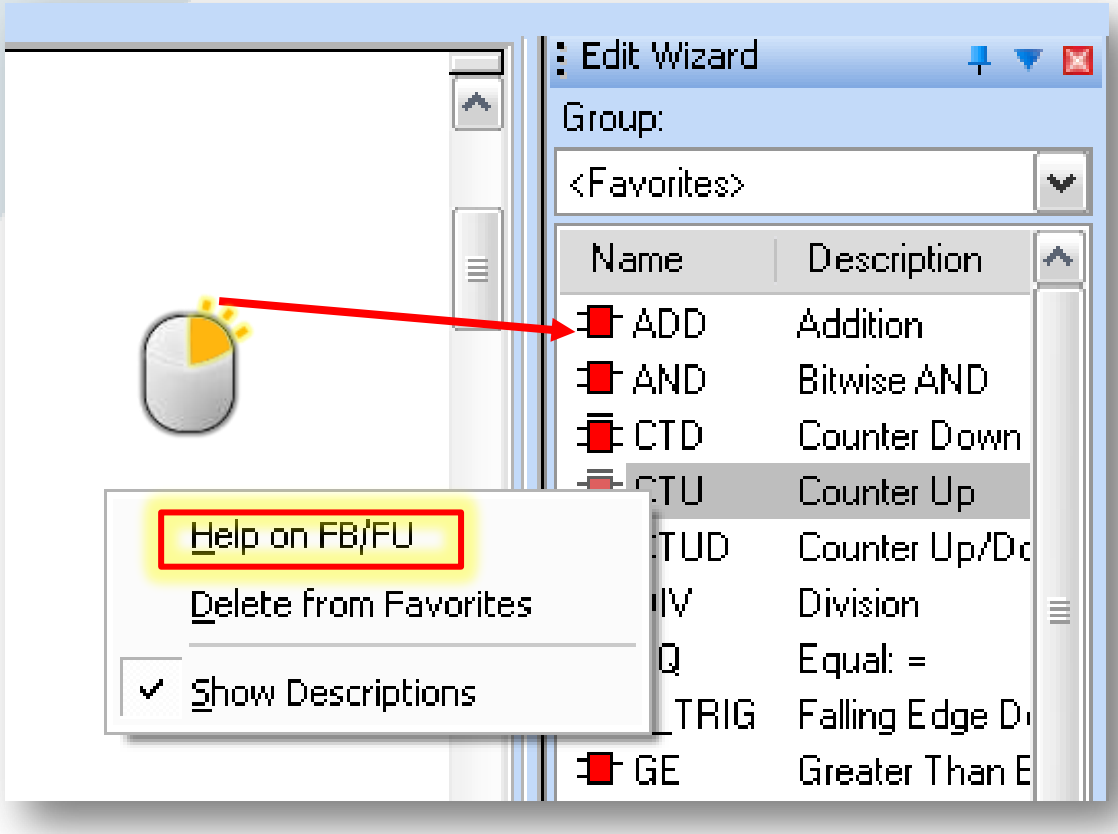




Help File
PLC_FBfun001.chm

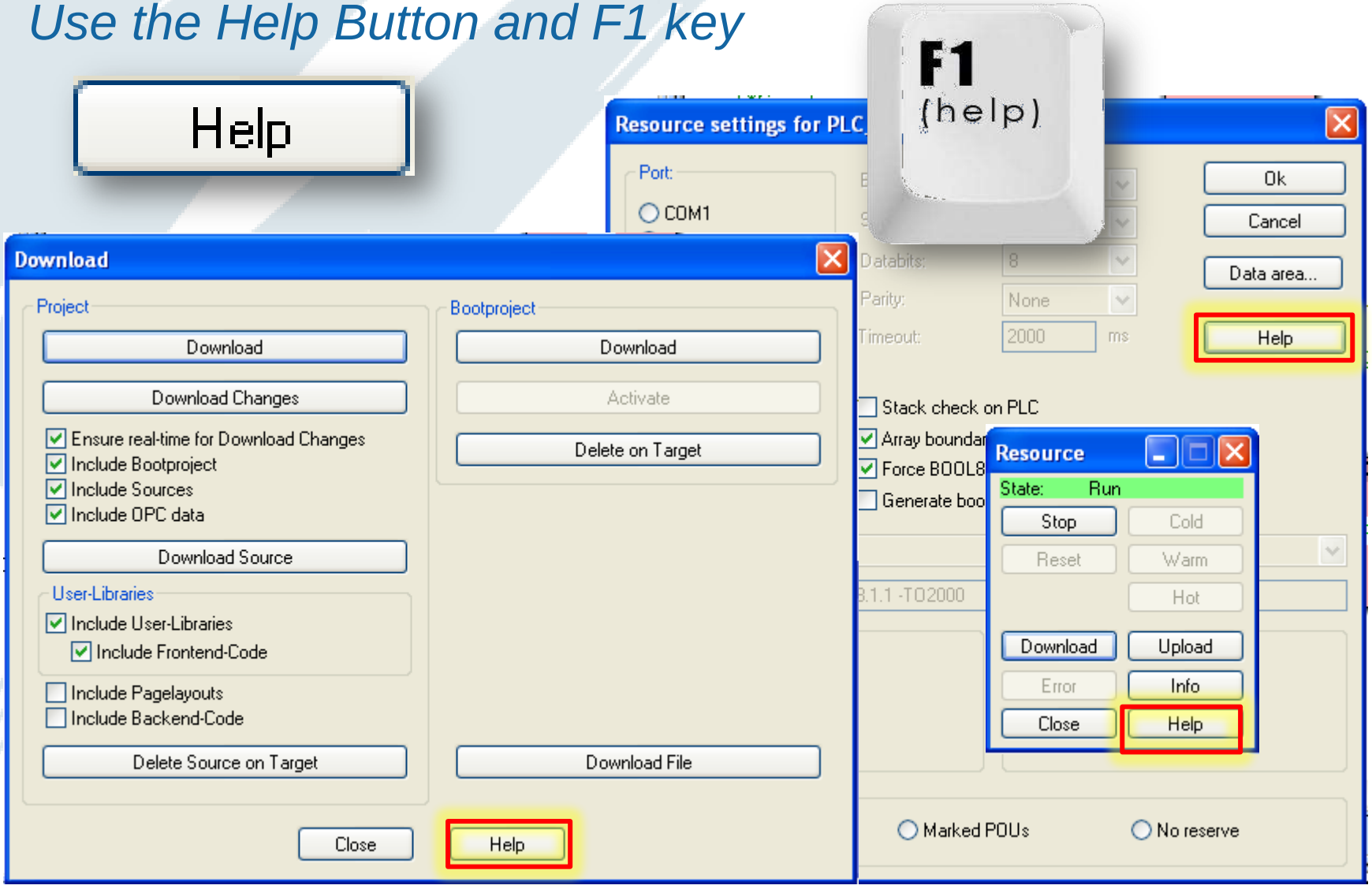
Right-Click Help

- *Right click a block for help*
  - *In the code*
  - *In the edit wizard*



Contextual Help Button

- *Use the Help Button and F1 key*



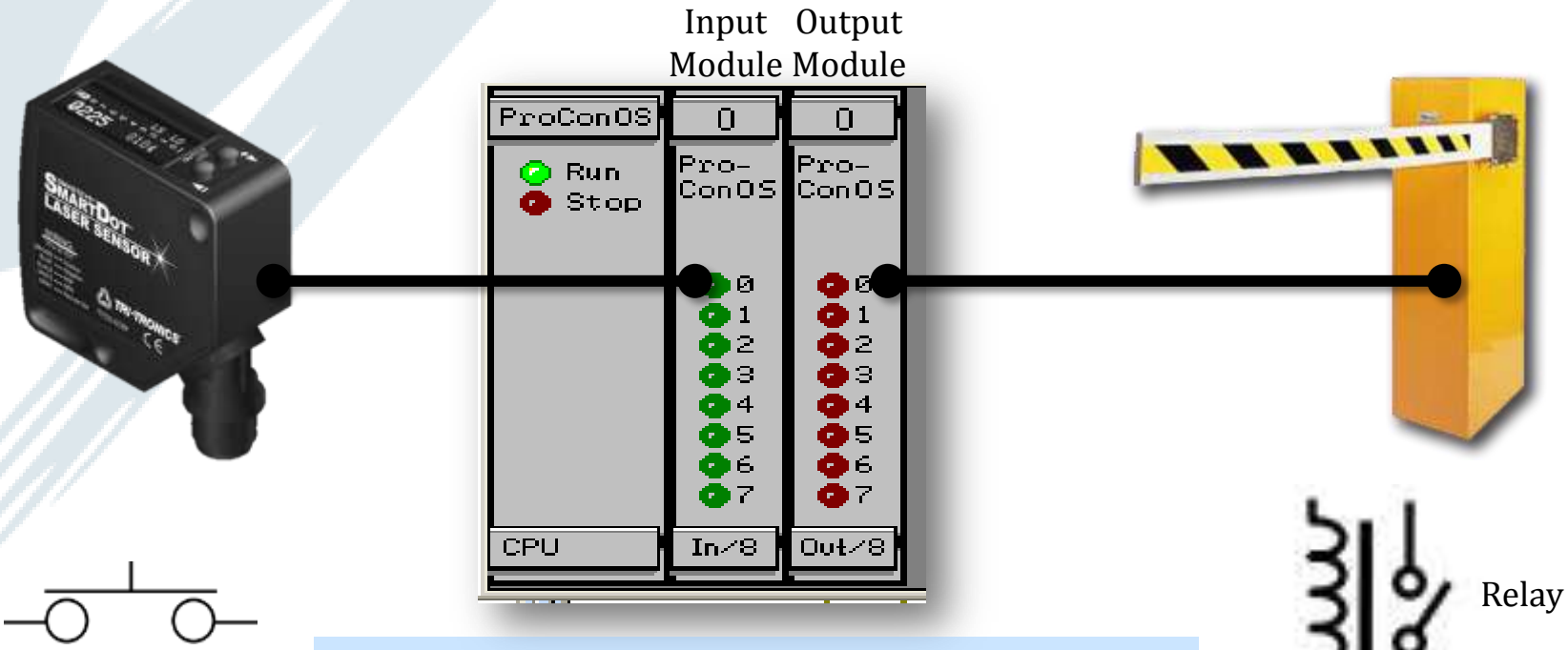
Using I/O

I/O Module Concept
I/O Variable Addressing
Configuration
Best Practice
Execution Sequence
Task Assignment

YASKAWA

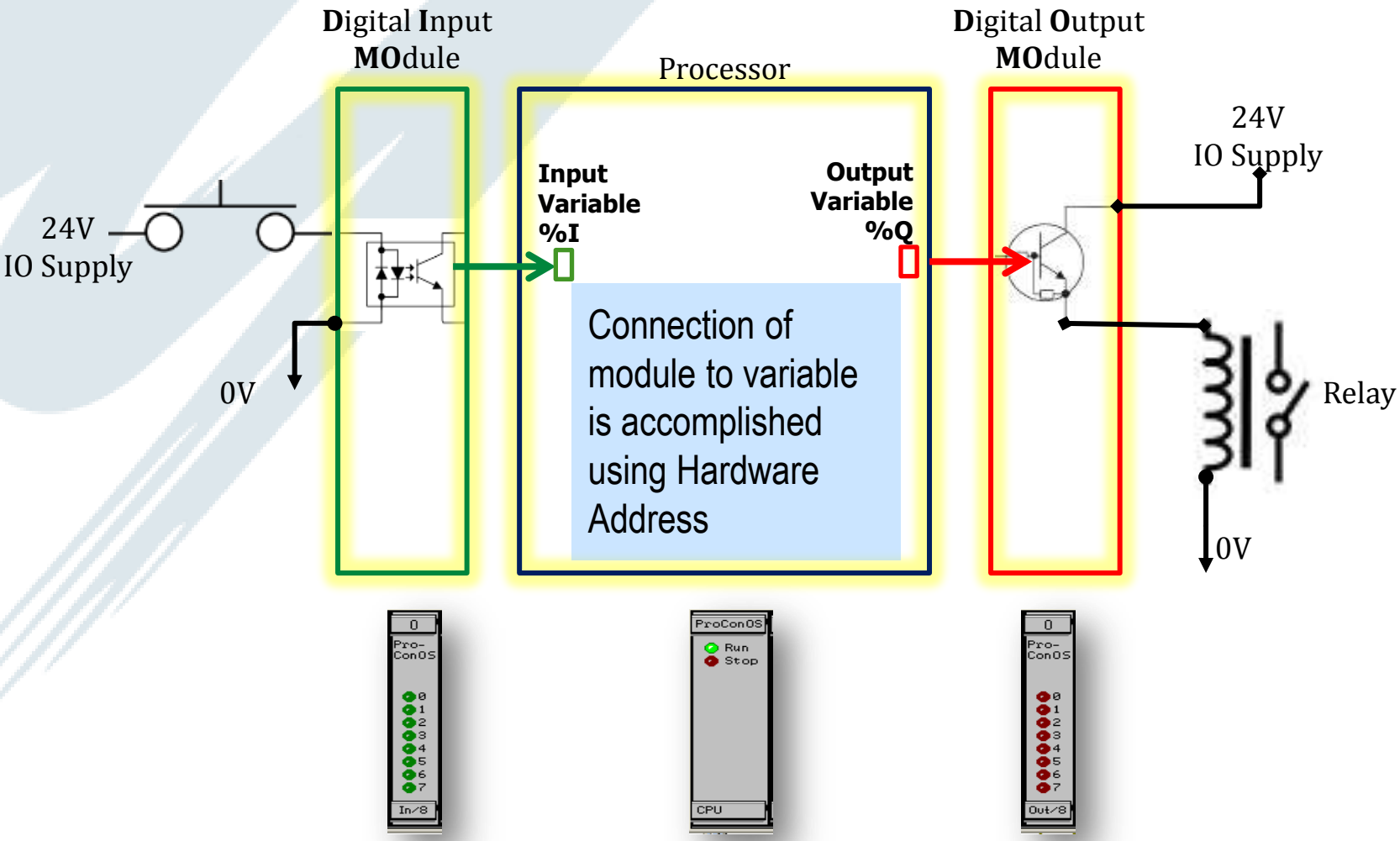
Field I/O Devices

- *Parking Gate Example Application*
  - *Connect physical sensor to PLC input*
  - *Connect physical gate to PLC output*



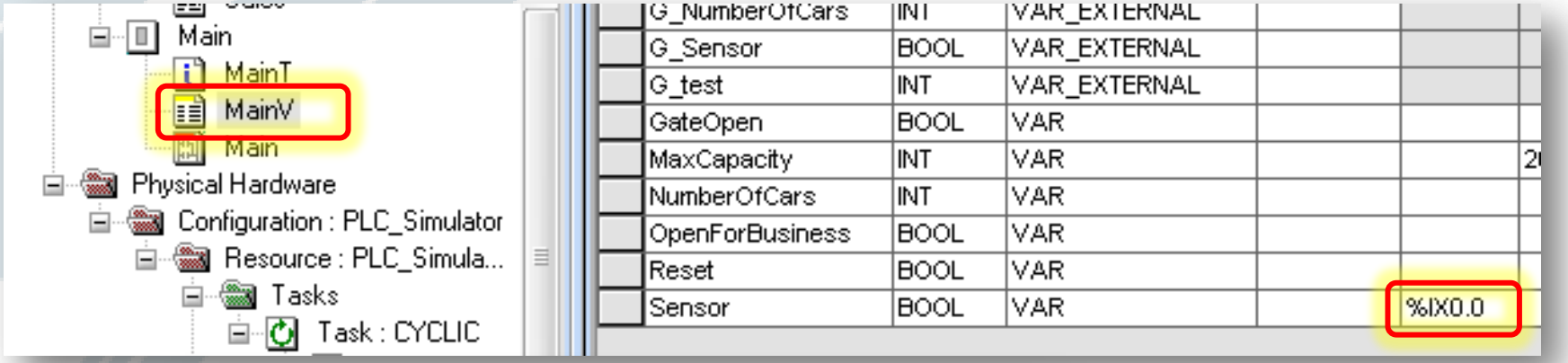
How can the PLC interface with these devices?

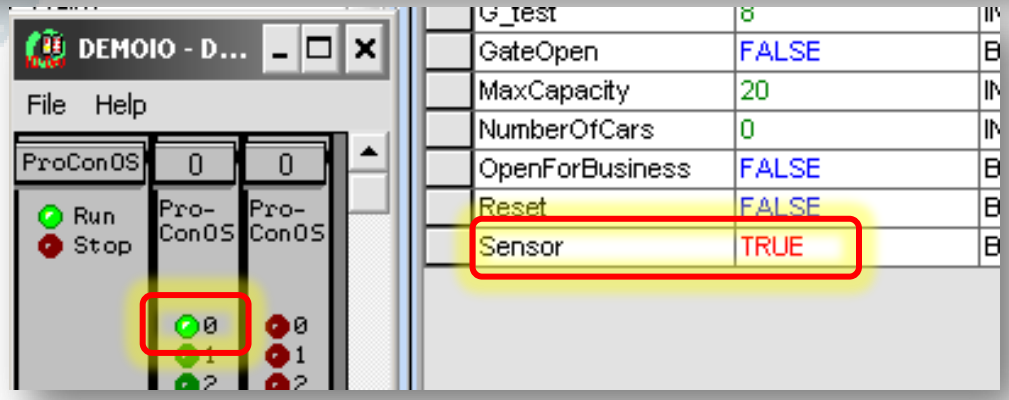
I/O Module Concept



Input Variable With Address

- *Any variable can be given a hardware address*
  - *Make "sensor' an input variable with address %IX0.0*
  - *In debug mode, turn the simulator input on/off*



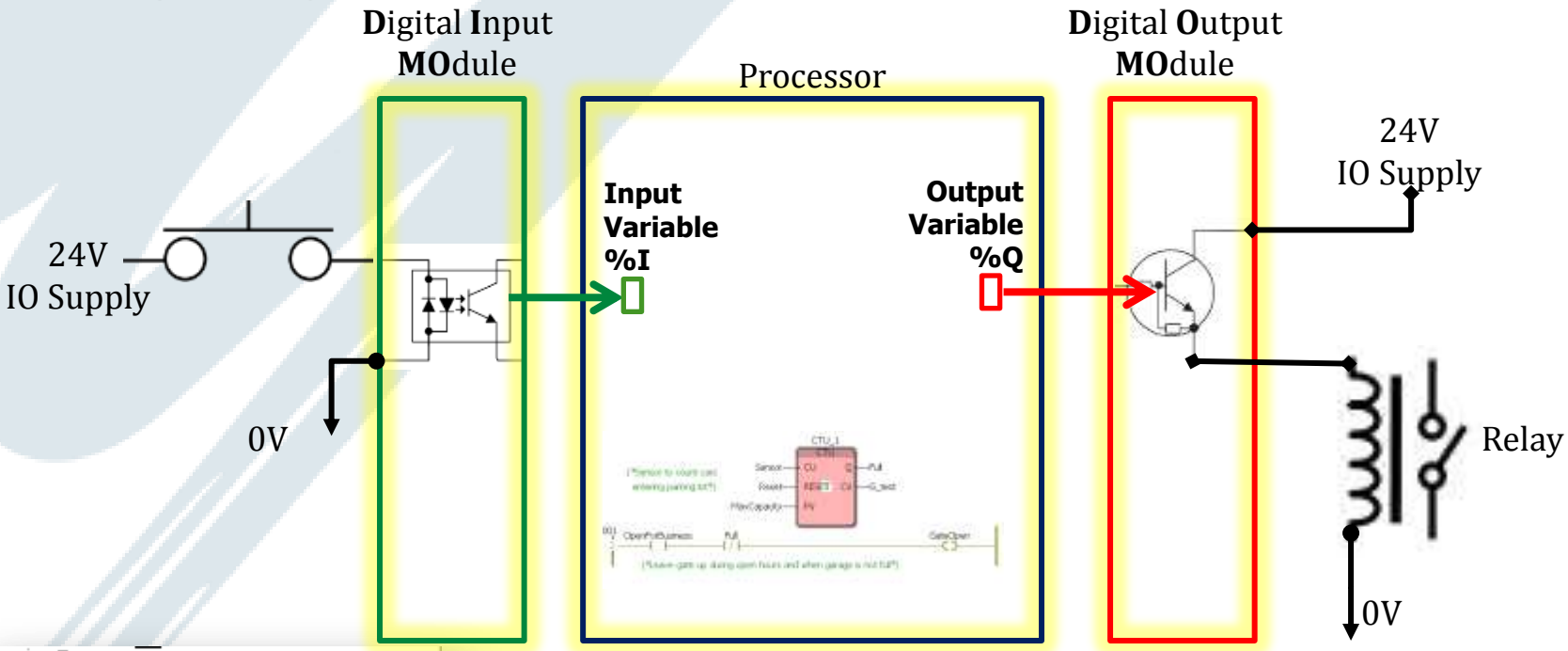


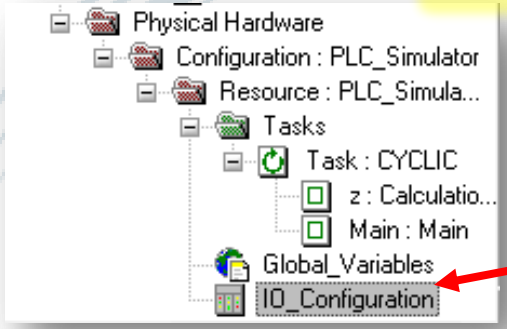
Where does %IX0.0 come from?

- Explanation coming next

IO Configuration

- *Connection of hardware to software*
  - *IO Driver*





I/O Addressing

- *IEC 61131-3 Byte level addressing*

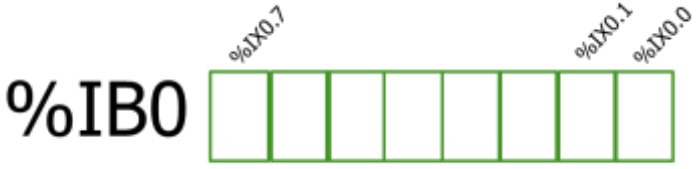
%IX 43210.7

Location Prefix
%I – Input Location
%Q – Output Location
%M – Memory Location

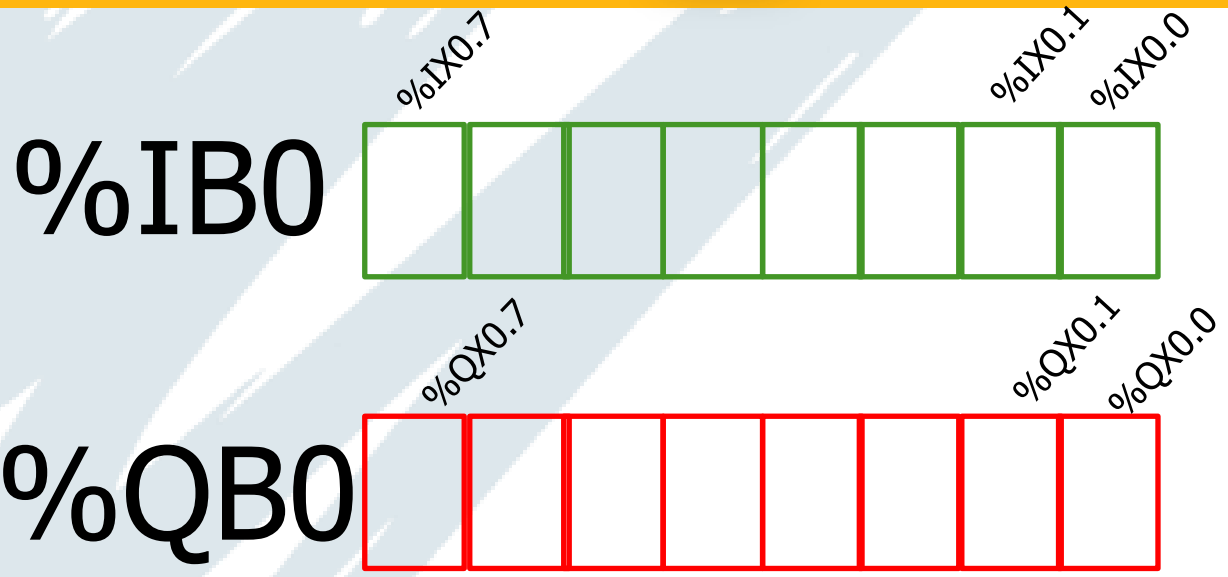
Data Size
X – Bit (1 bit)
B – Byte (8 bits)
W – Word (16 bits)
D – Double (32 bits)
L – Long (64 bits)

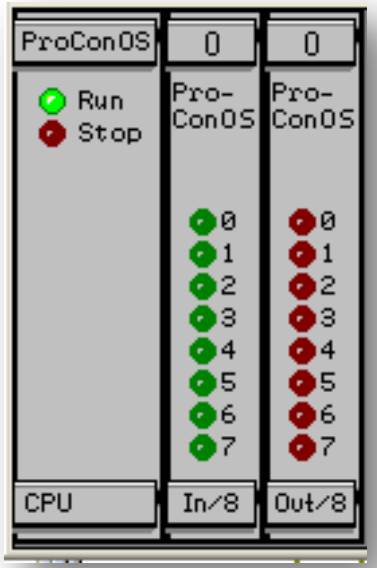
Memory Address
0 to max supported
Byte level addressing

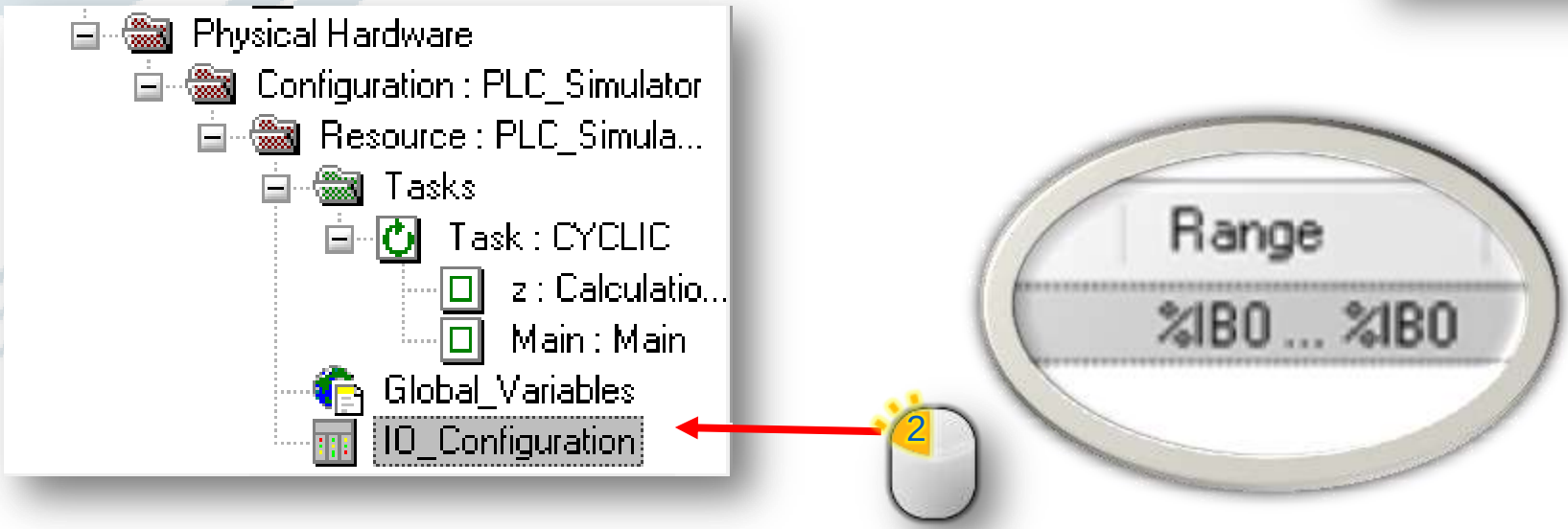
Bit Indicator
.0 to **.7**
Only for Bit (X) data size



IO as a Byte

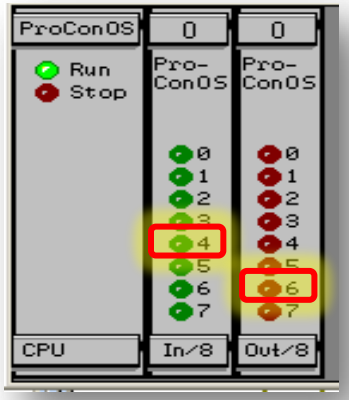

%IB0
%IX0.7
%IX0.1
%IX0.0
%QB0
%QX0.7
%QX0.1
%QX0.0

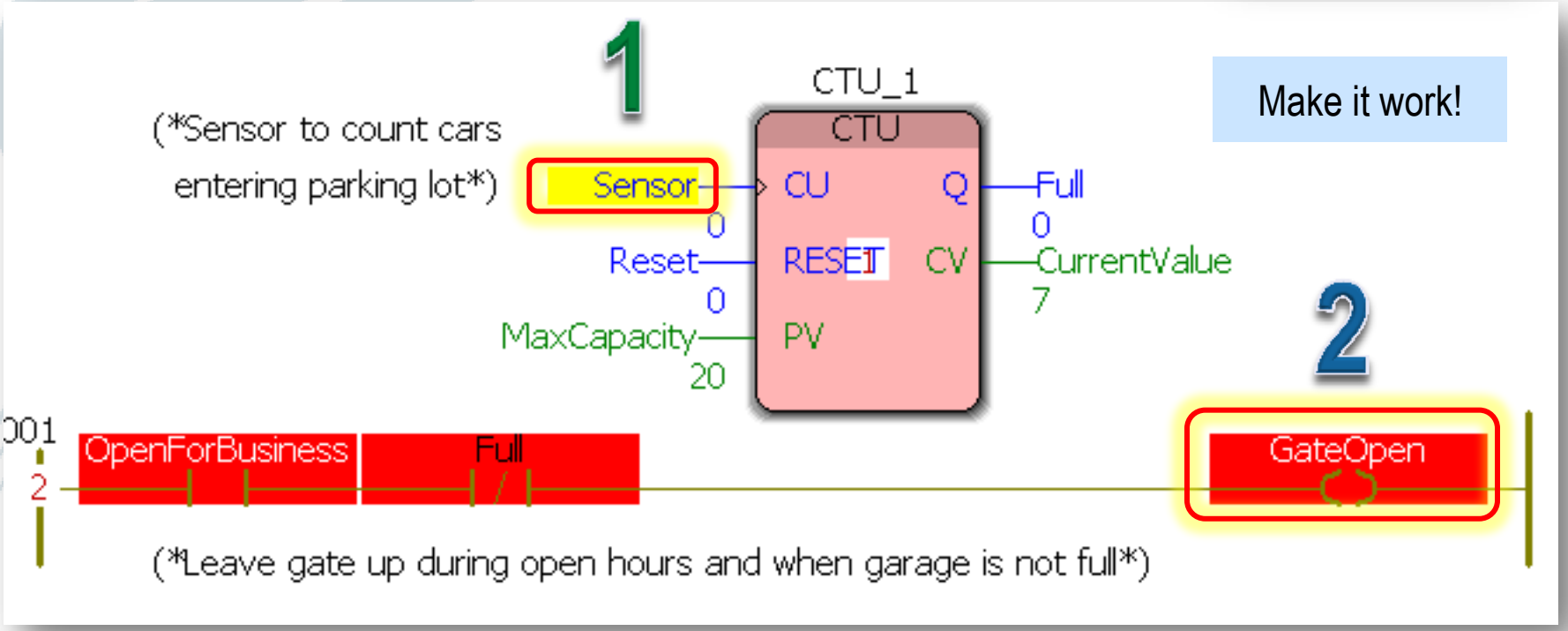

ProConOS
0
0
Run
Stop
Pro-
ConOS
Pro-
ConOS
0
1
2
3
4
5
6
7
CPU
In/8
Out/8


Physical Hardware
Configuration : PLC_Simulator
Resource : PLC_Simula...
Tasks
Task : CYCLIC
z : Calculatio...
Main : Main
Global_Variables
IO_Configuration
2
Range
%IB0 ... %IB0

Quick Quiz

1. **Sensor** *corresponds to input 4 on the simulator. What address is required?*
   - *%IX0.4*
2. **GateOpen** *corresponds to output 6 on the simulator. What address is required?*
   - *%QX0.6*

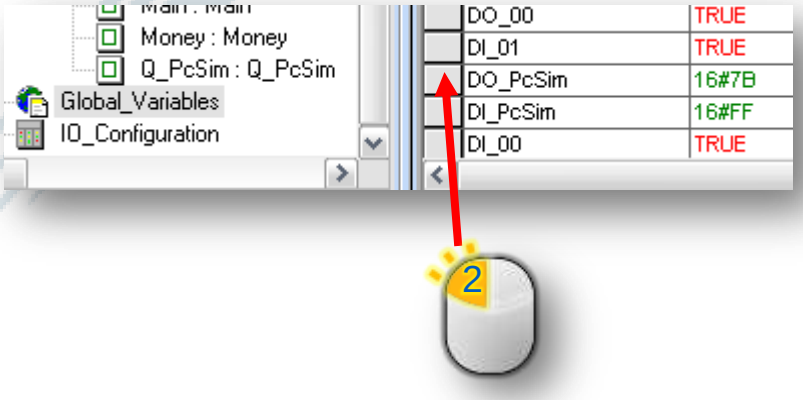


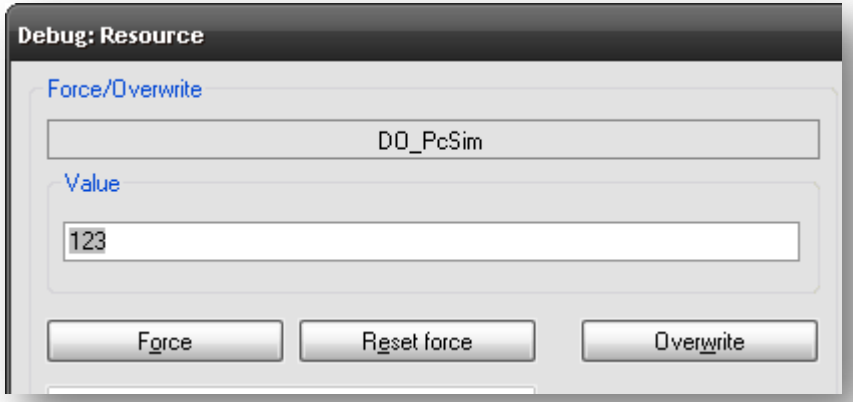


IO as a Byte

- *Create an input variable* **DI_PcSim** *in Global_Variables*
  - *All 8 inputs as a byte*
- *Create output variable* **DO_PcSim** *in Global_Variables*
  - *All 8 outputs as a byte*
- *Read inputs in debug mode*
  - *What is the value with all inputs on?*
- *Overwrite outputs in debug mode*
  - *How can you enter hex values directly?*

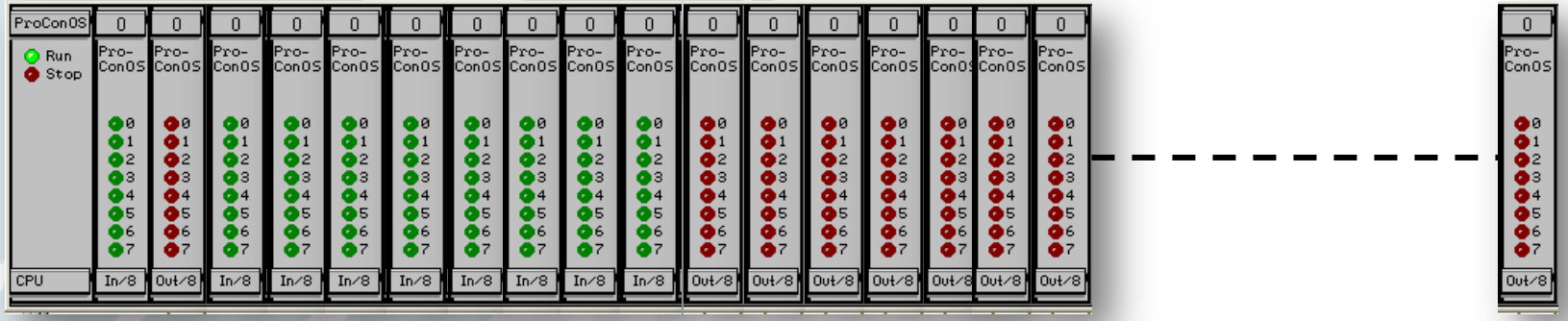
What happens in debug mode if there is no IO module configured at the address of the variable – for example try %IB2

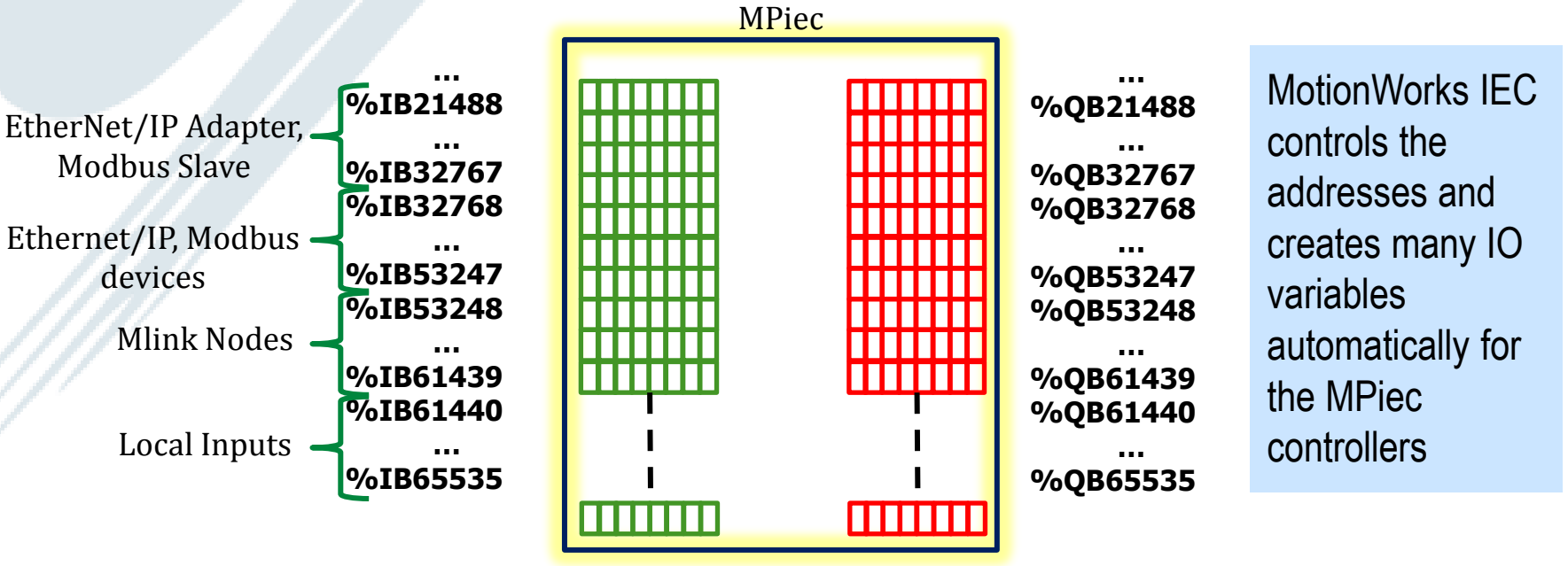




MPiec Addressing

- *Large number of input modules possible in real PLC*

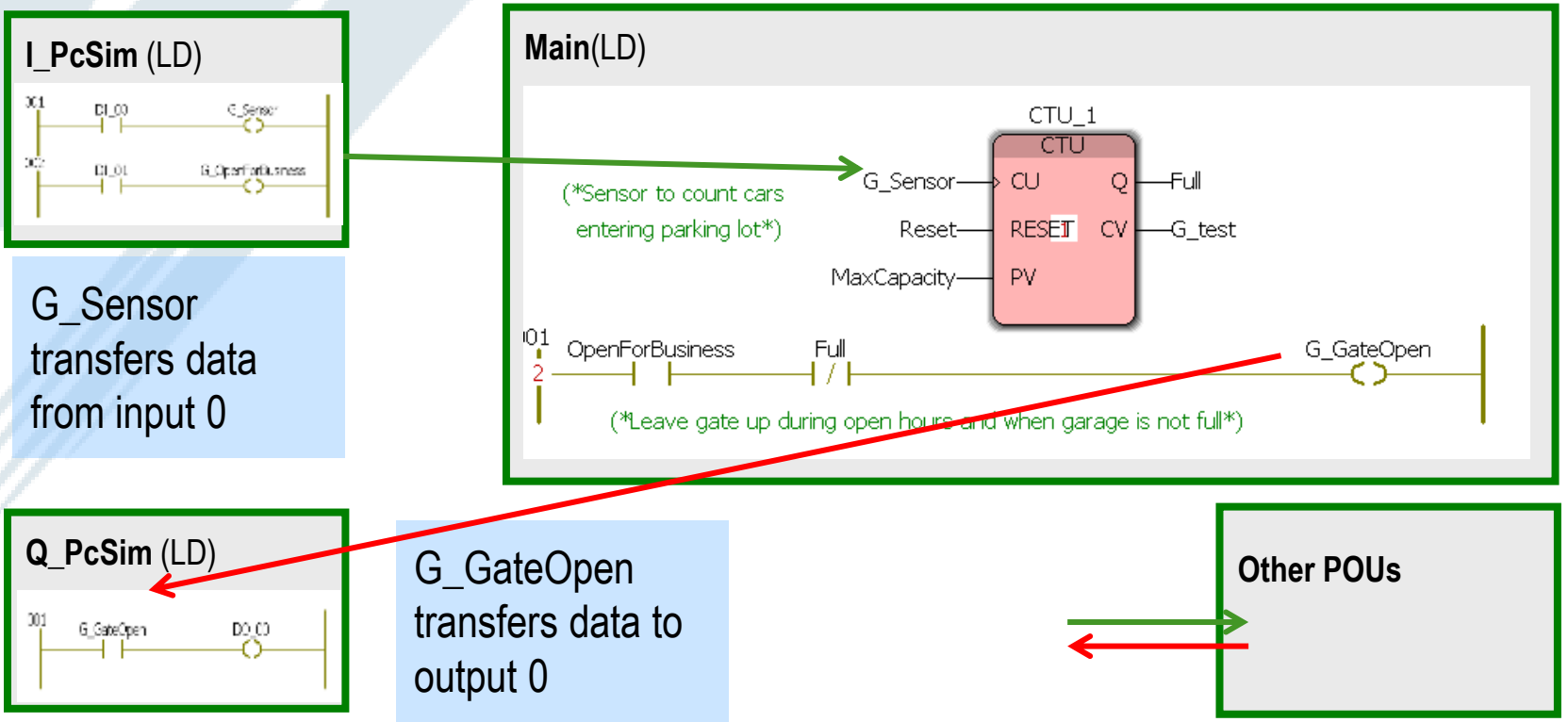




MotionWorks IEC controls the addresses and creates many IO variables automatically for the MPiec controllers

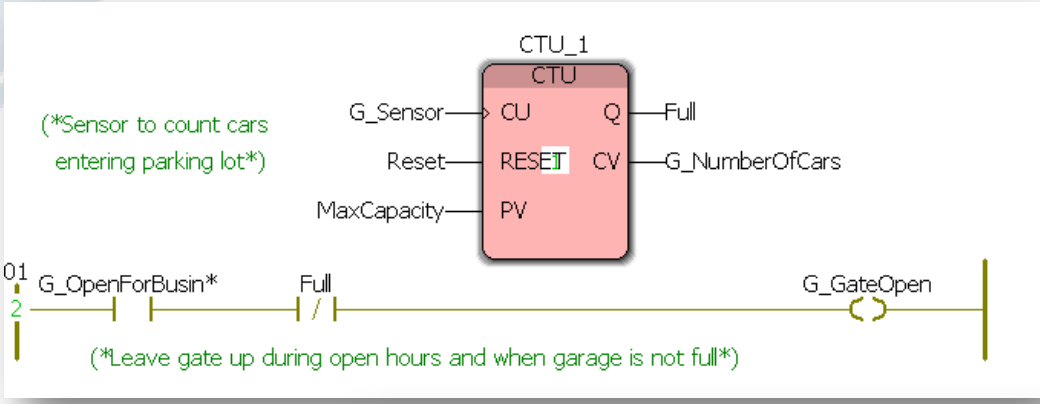
IO Programming Best Practice

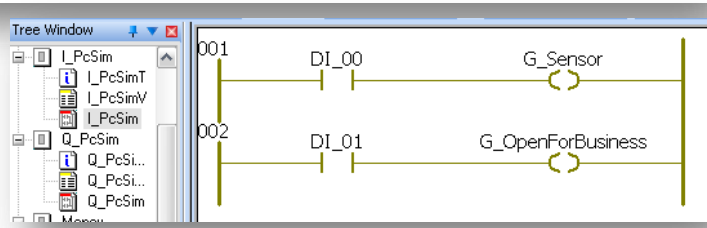
- *Generic IO variables in Global_Variables list*
  - *DI_00, DI_01, etc.*
  - *DO_00, DO_01, etc.*
- *Program IO variables in separate program POUs*
  - *POUs easily reused with other IO hardware – modularity*
    - *Example: move from one HMI to another – addresses may change*
  - *1 or 2 POUs affected by hardware change*
  - *Transfer I/O data through global variables*

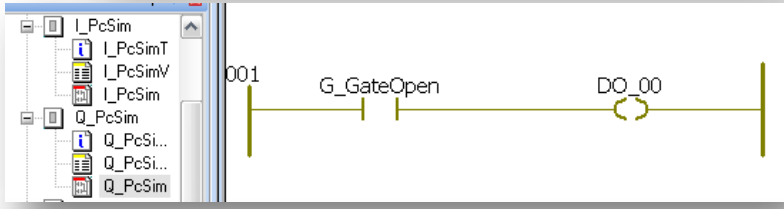


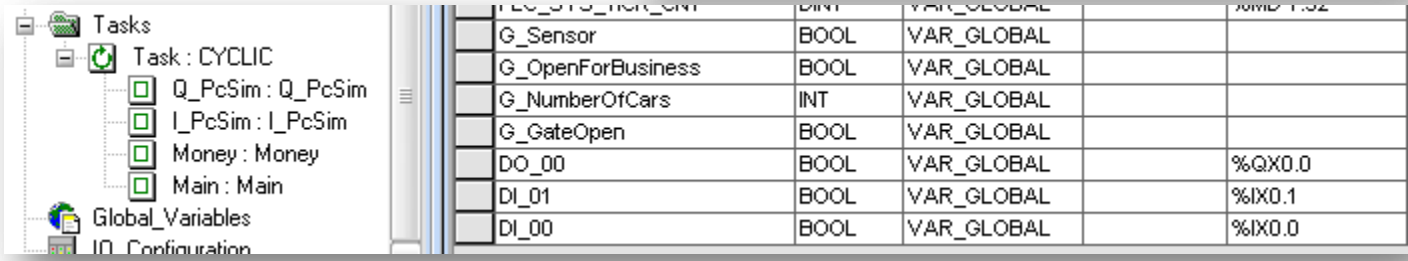
Best Practice Project Update

- *Project Update Checklist*
  - *Global Variables in Code*
    - » *G_Sensor*
    - » *G_OpenForBusiness*
    - » *G_GateOpen*
  - *Global IO Variables*
    - » *DI_00, DI_01*
    - » *DO_00*
  - *LD Program POUs*
    - » *Q_PcSim*
    - » *I_PcSim*
    - » *Instance in Task*



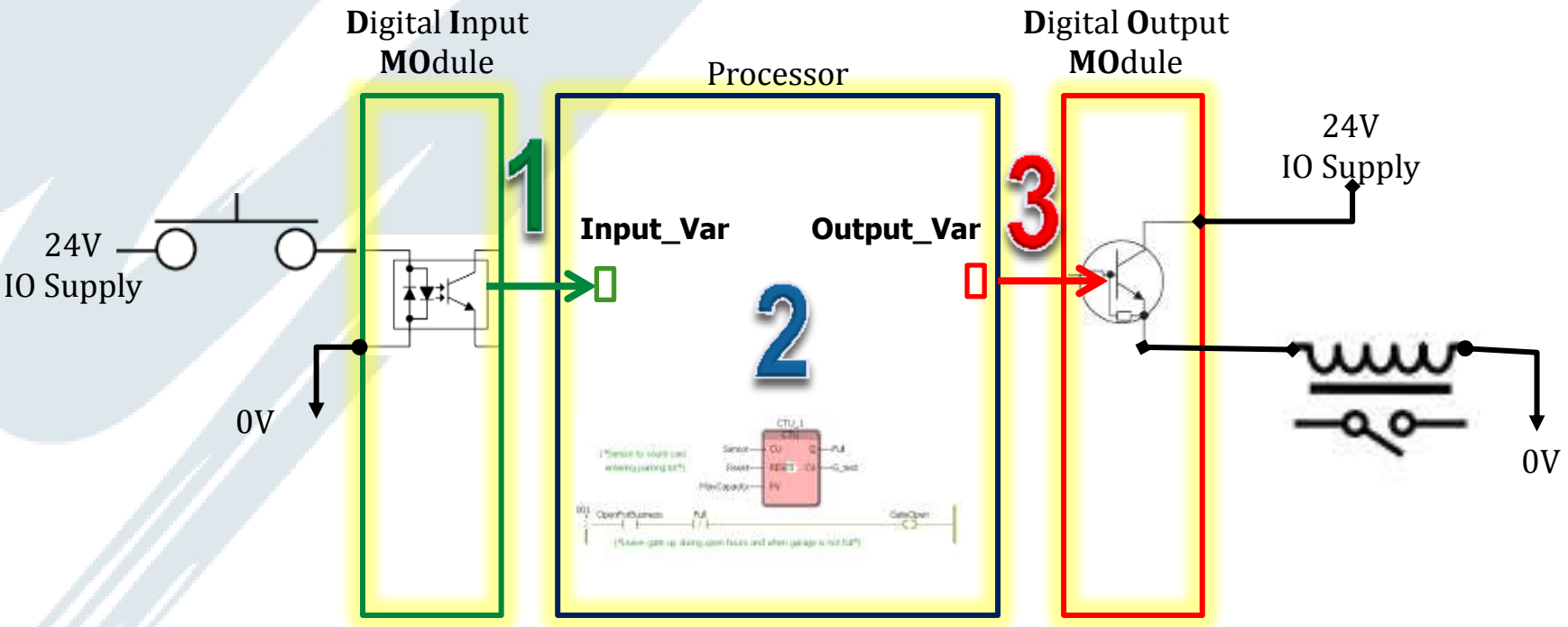






G_Sensor	BOOL	VAR_GLOBAL		
G_OpenForBusiness	BOOL	VAR_GLOBAL		
G_NumberOfCars	INT	VAR_GLOBAL		
G_GateOpen	BOOL	VAR_GLOBAL		
DO_00	BOOL	VAR_GLOBAL		%QX0.0
DI_01	BOOL	VAR_GLOBAL		%IX0.1
DI_00	BOOL	VAR_GLOBAL		%IX0.0

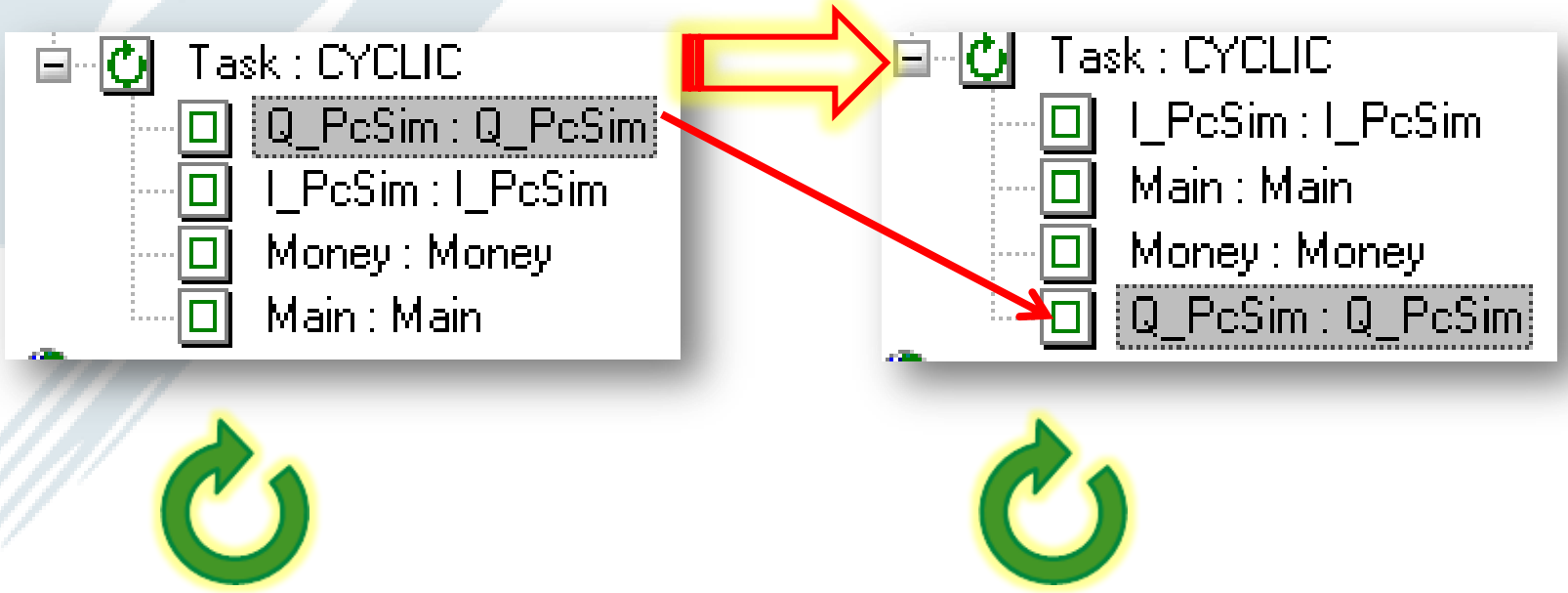
Execution Sequence



1. Input variables are read from input modules
2. Processor solves application code, updating output variables from top to bottom of task
3. Output variables are written to output modules

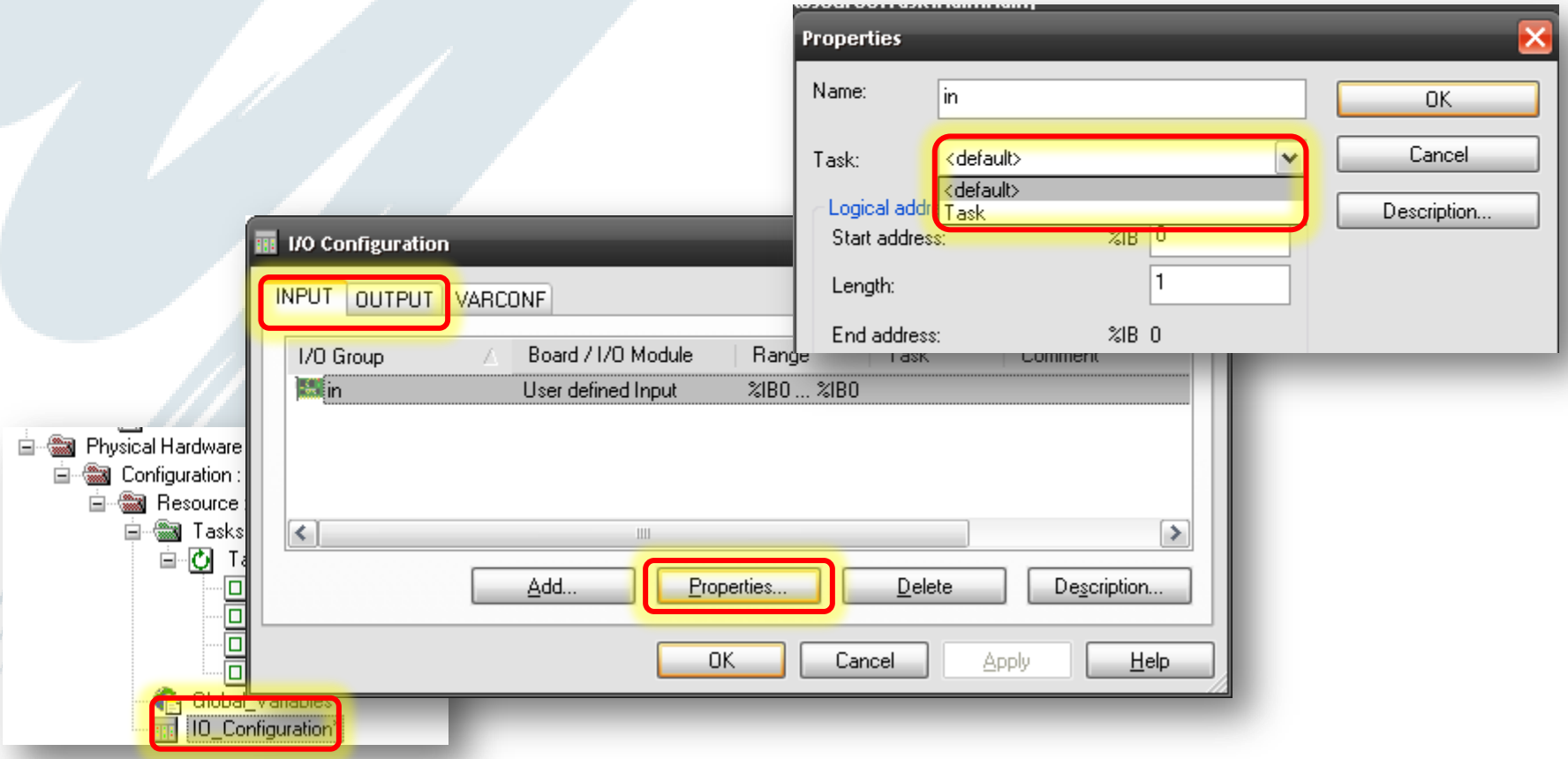
IO In Task List

- *Order of POU instances in task affect operation*
  - *Read Input Variables to global variables at top of task list*
  - *Write Output Variables at bottom of task list*

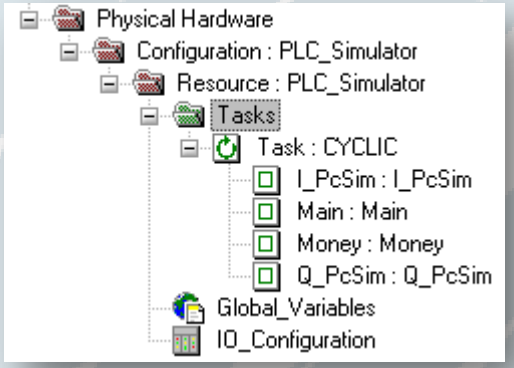


IO Task Assignment

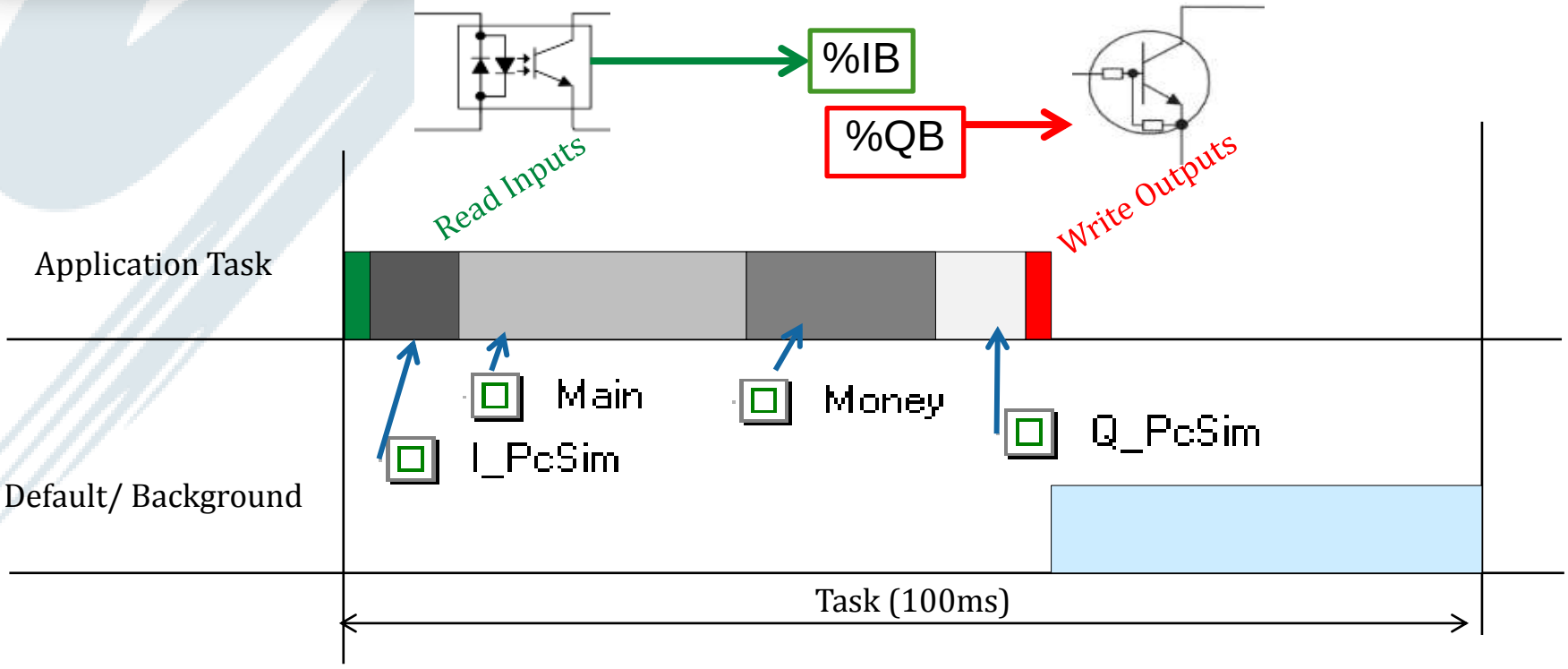
- *Assign IO to the task in which they are used*
  - *Physical Hardware – IO_Configuration: Properties - task*



IO Execution Timing



- *Inputs are read into %IB before task execution starts*
- *Task is executed*
- *%QB are written to outputs immediately after task execution ends.*



Monitoring

Force I/O variables

LREAL Value Display

Watch Window

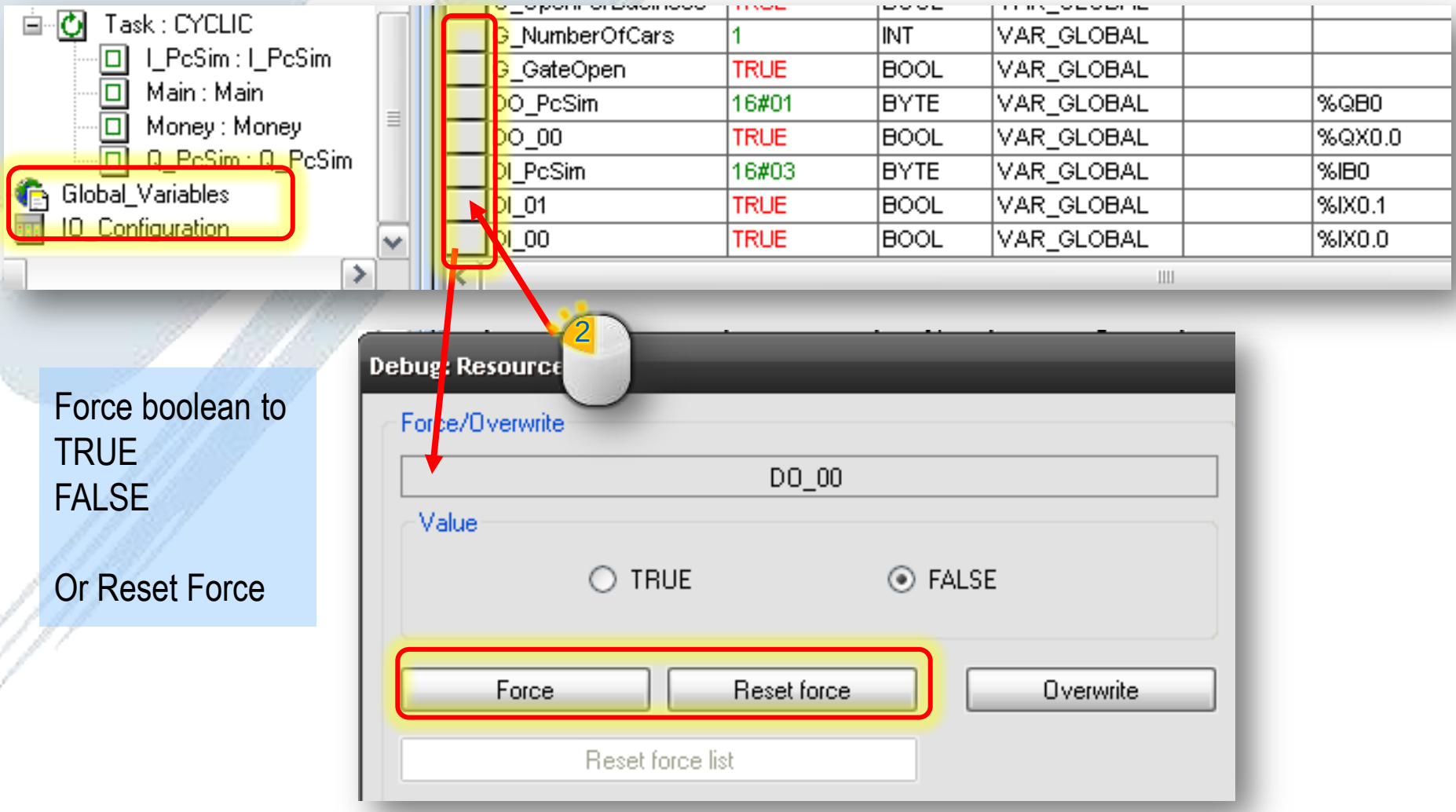
Tooltip

Cross Reference

YASKAWA

Force I/O Variables

Monitor Global Variables

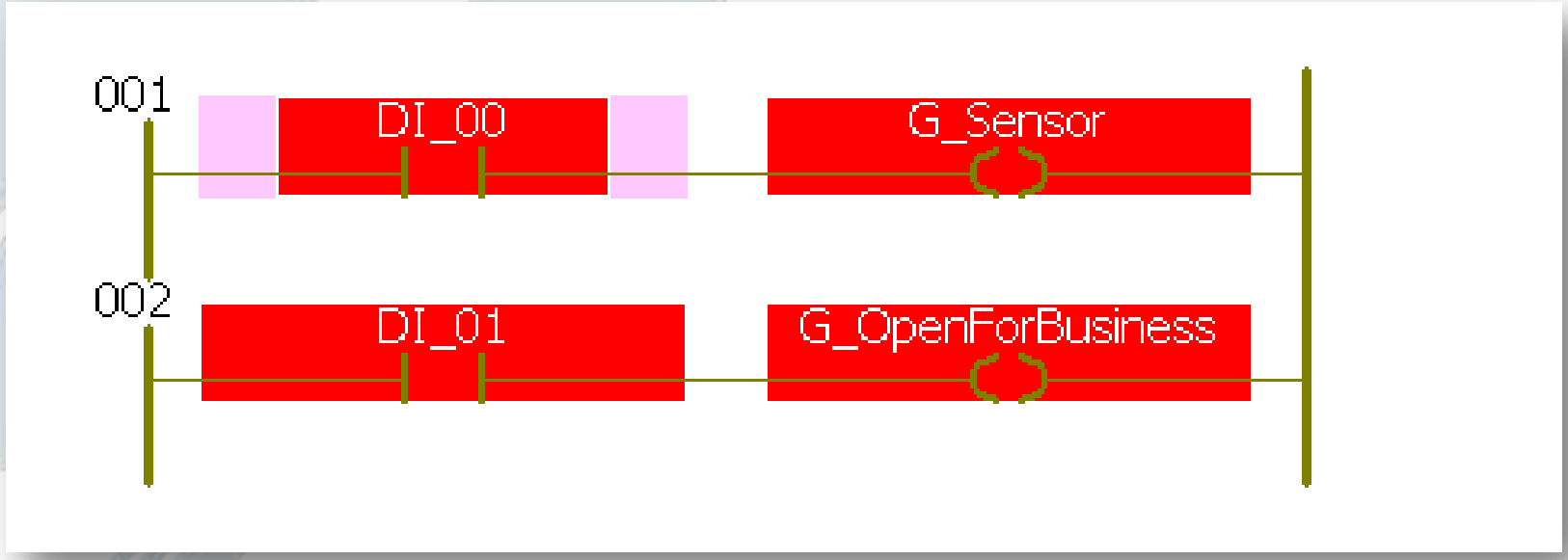


Force boolean to
TRUE
FALSE

Or Reset Force

Force Status

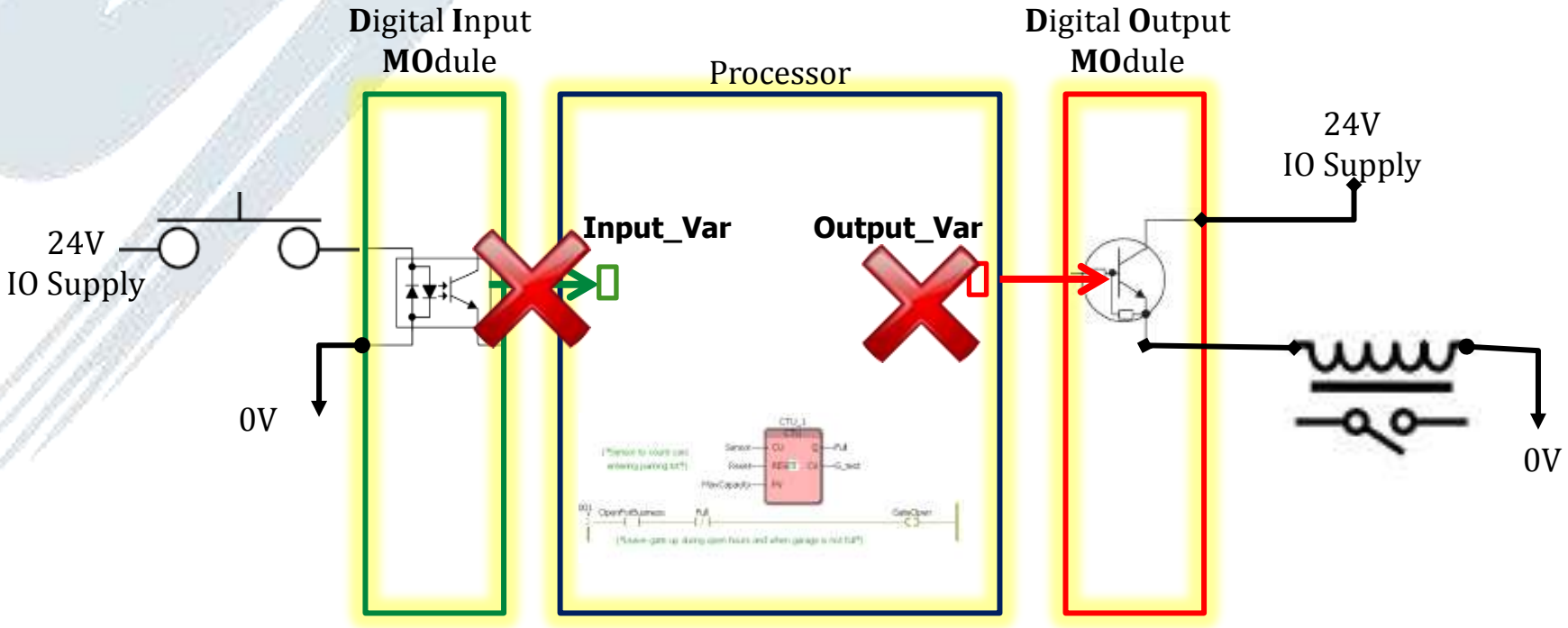
- *Pink Highlight*



	DO_PcSim	16#00	BYTE	VAR_GLOBAL		%QB0
	DO_00	FALSE	BOOL	VAR_GLOBAL		%QX0.0
	DI_PcSim	16#7B	BYTE	VAR_GLOBAL		%IB0
	DI_01	TRUE	BOOL	VAR_GLOBAL		%IX0.1
	DI_00	TRUE	BOOL	VAR_GLOBAL		%IX0.0

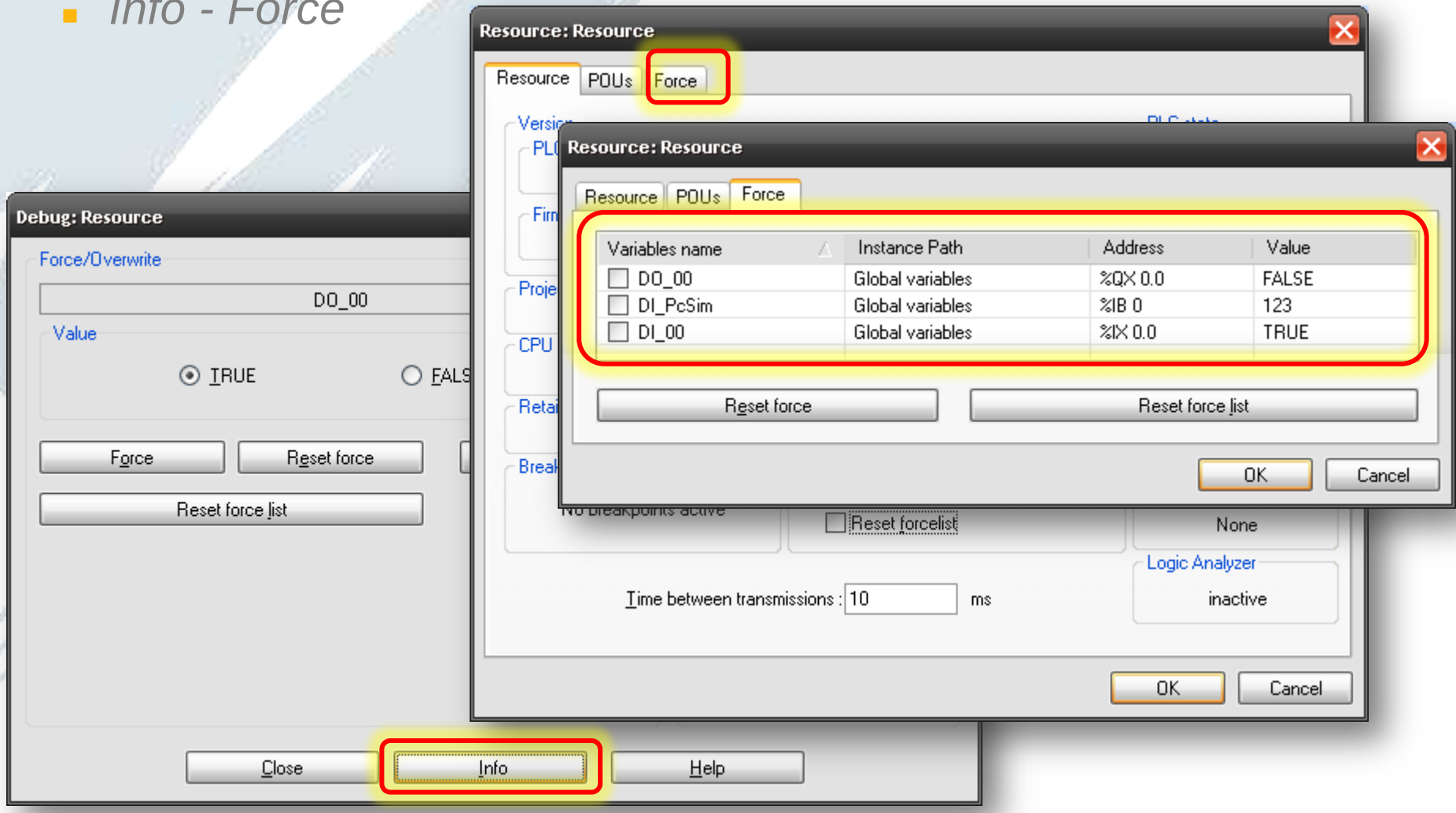
Force I/O Variables

- *What is "Forcing" and an I/O variable?*
  - *Variable in processor is held at a specific value regardless of code execution*
  - *Processor does not see true input state*
  - *Output does not reflect result of code*



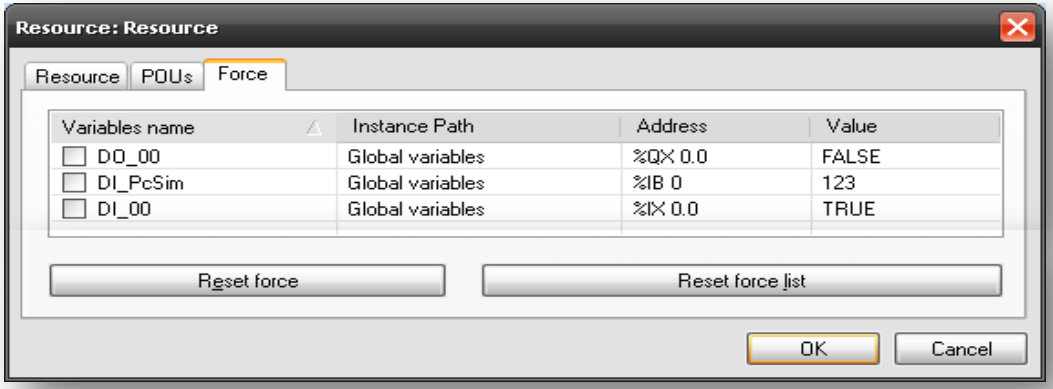
Force List

- *Debug Mode*
  - *Info - Force*



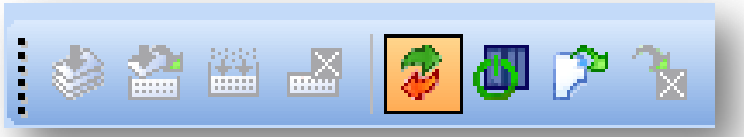
PLC Force State

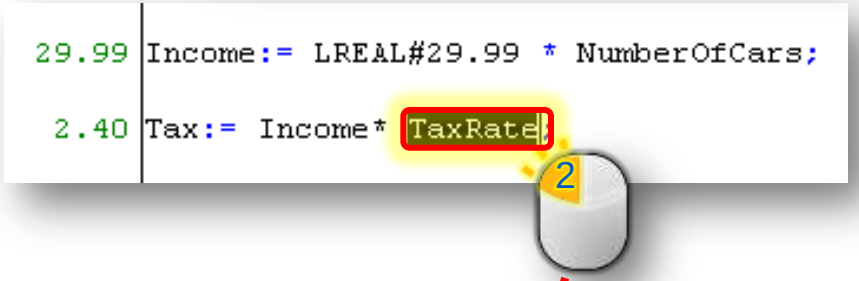
- *Limitations during Force*
  - *Force state is not saved in the project*
  - *Force DOES survive warm start and cold start*
  - *Task instances cannot be deleted or rearranged*
  - *Task instances CAN be copied*
    - » *Click and drag will create a copy*
    - » *Best not to edit code during force*

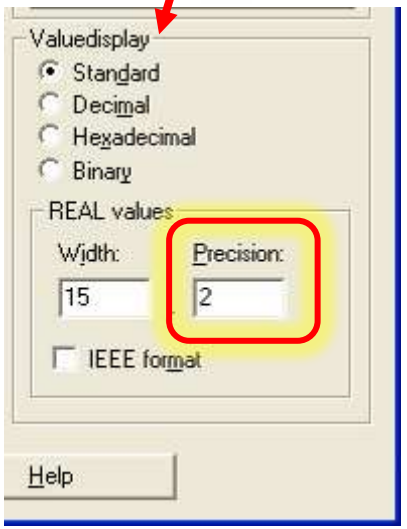


LREAL Value Display

- *Add line to "Money" POU*
  - *Tax:= Income* TaxRate;*
- *Value Display*
  - *Debug Mode*
  - *Double click any variable*
  - *Adjust Precision*
  - *Close*

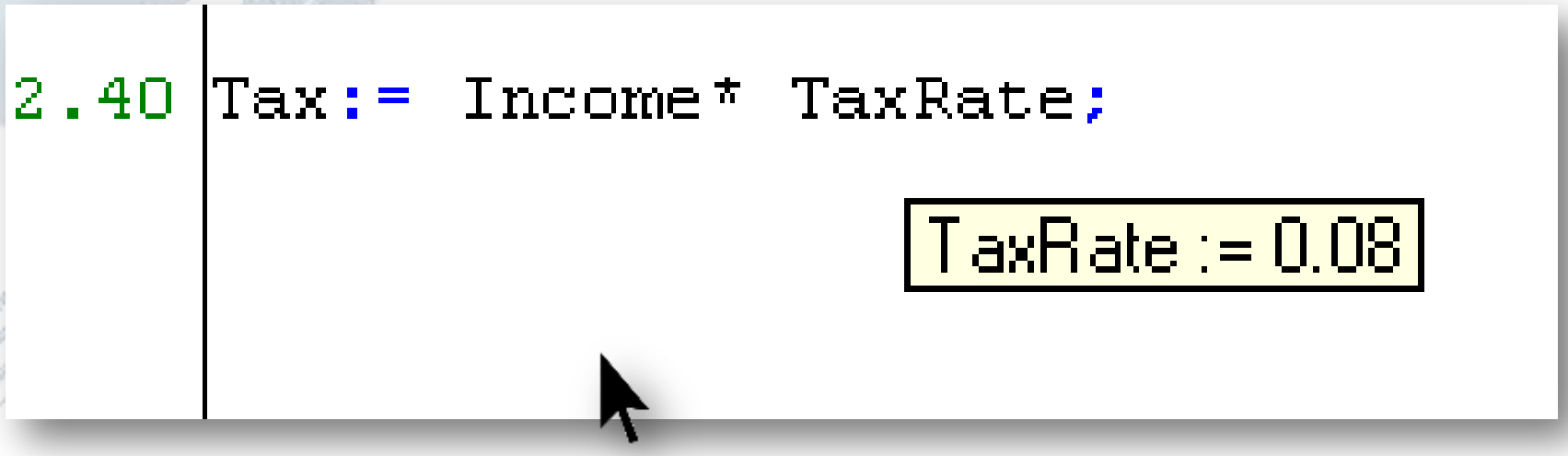






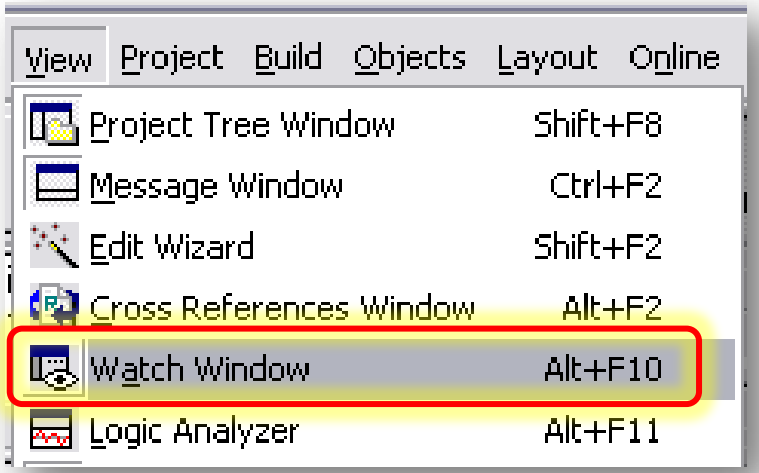
Tooltip

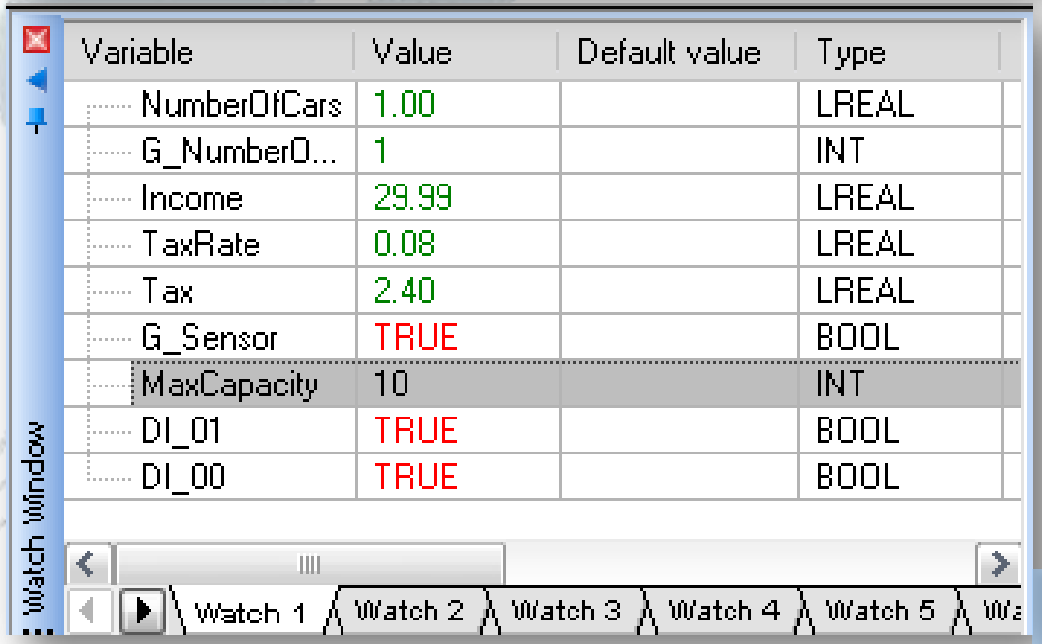
- *Tooltip shows data of variable*
  - *Debug Mode*
  - *Hover over variable*
- *Especially useful in ST code*



Watch Window

- *Display Watch Window*
  - *Debug*
  - *View – Watch Window*



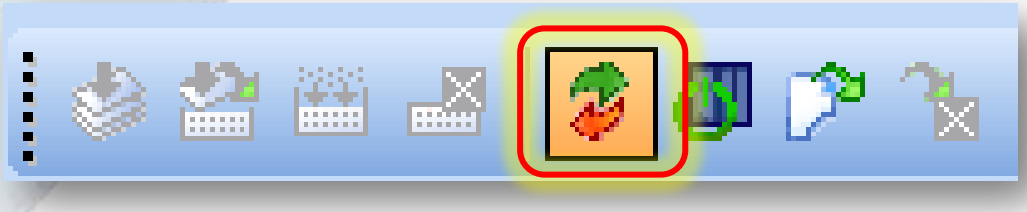


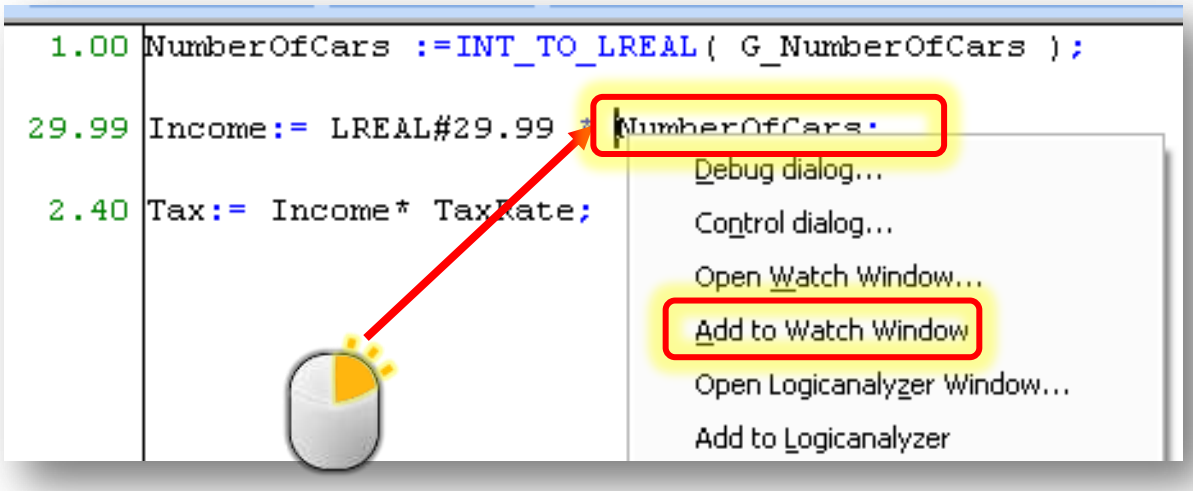
The Watch Window is only available in DEBUG MODE

Tab names can be changed

Watch Window

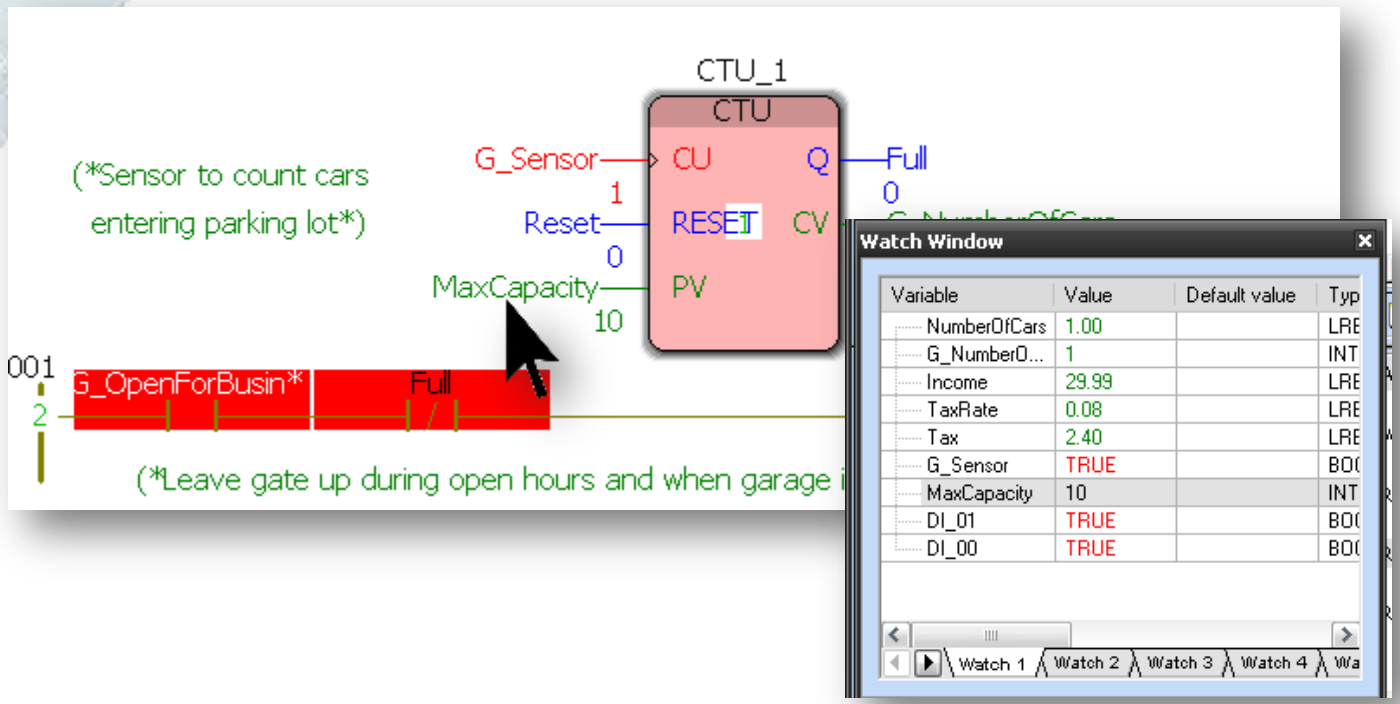
- *Right-Click Method*
  - *Debug ON*
  - *Right-Click Variable in Code*
  - *Right-Click Variable in Variable List*
  - *Choose "Add to Watch Window"*





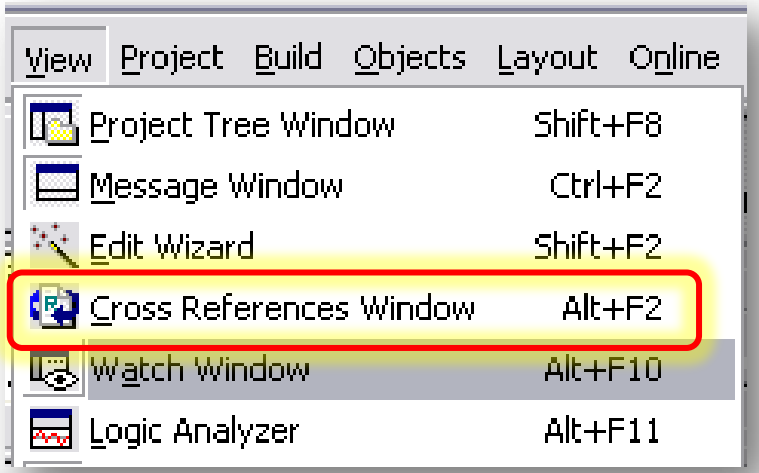
Watch Window

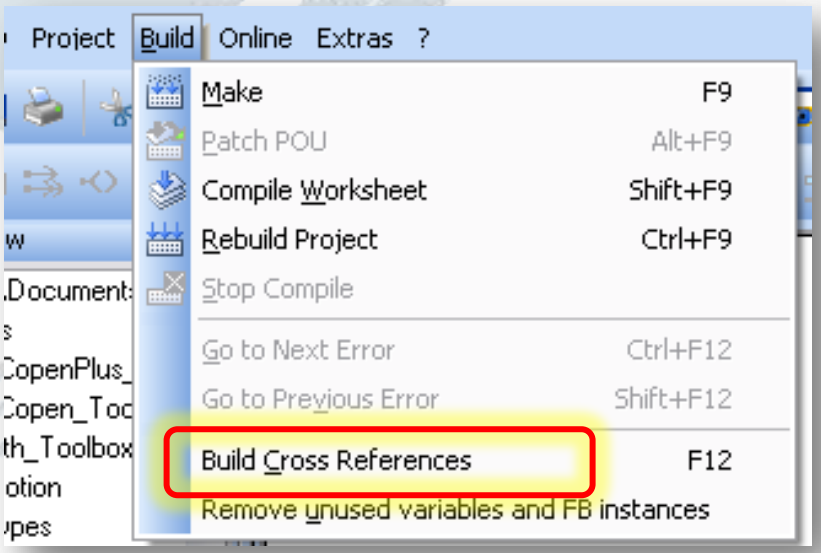
- *Drag & Drop Method*
  - *Debug ON*
  - *Drag and drop from code only*
    - » ***Variable List Not supported***

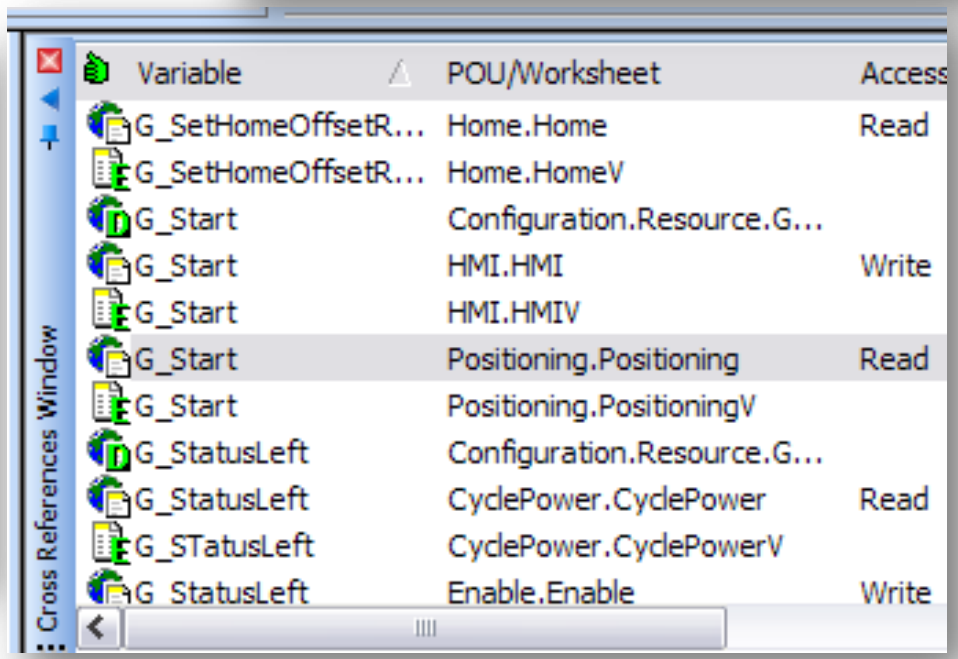


Cross Reference

- *Display Cross Reference*
  - *View – Cross References Window*
- *Build Cross Reference*
  - *Build – Build Cross References*

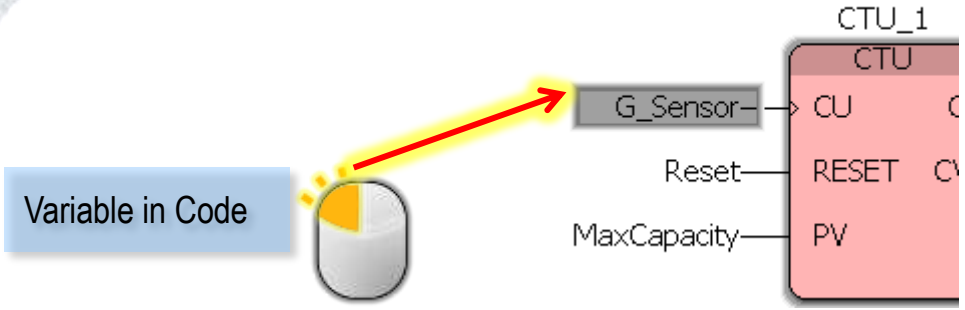


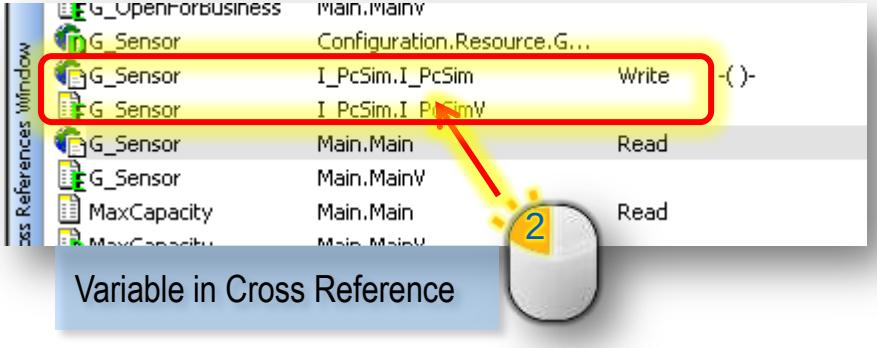


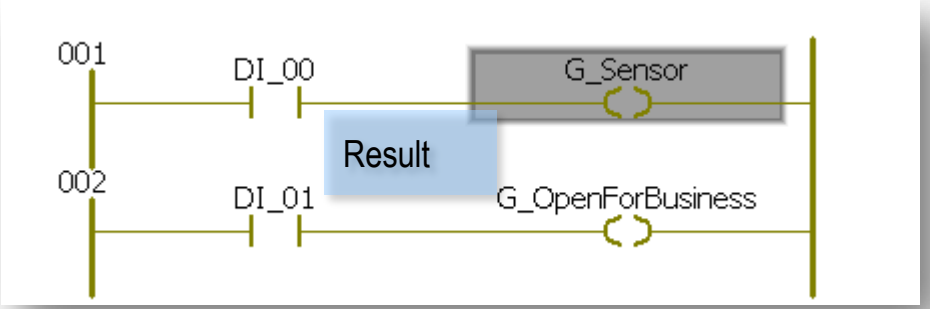


Cross Reference

- *Navigation*
  - *Click on variable in code*
    - » *Cross reference window updates*
    - » *Read or Write access indicated*
  - *Double-Click to open variable from Cross Reference*
    - » *Code worksheet opens*

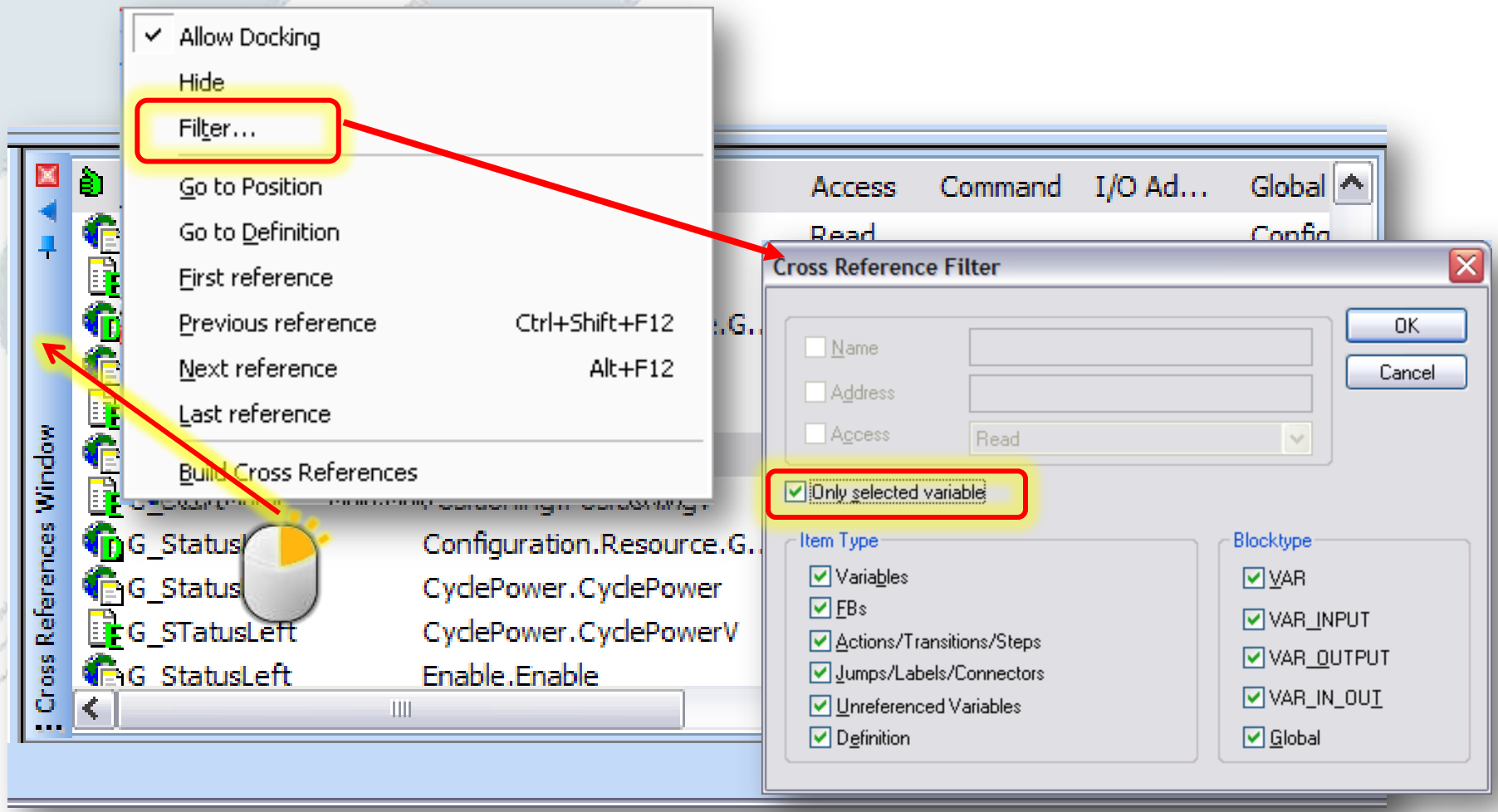






Cross Reference

- *Filter*
  - *Right-Click (title bar) – Filter*



Driving Value

IEC 61131-3 Programming Overview

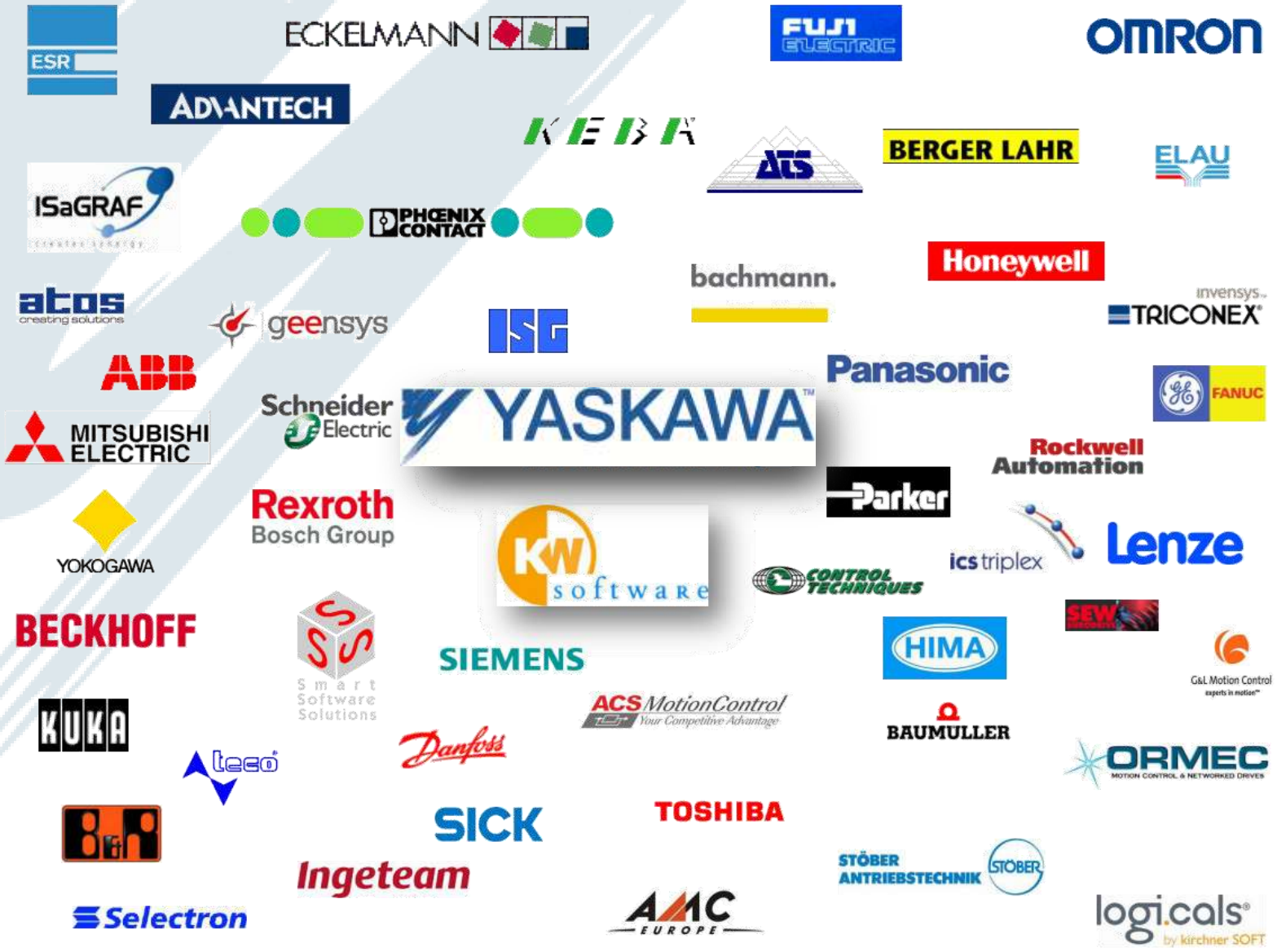
Standard
Languages
Data Types
Software Model
Task Execution

YASKAWA

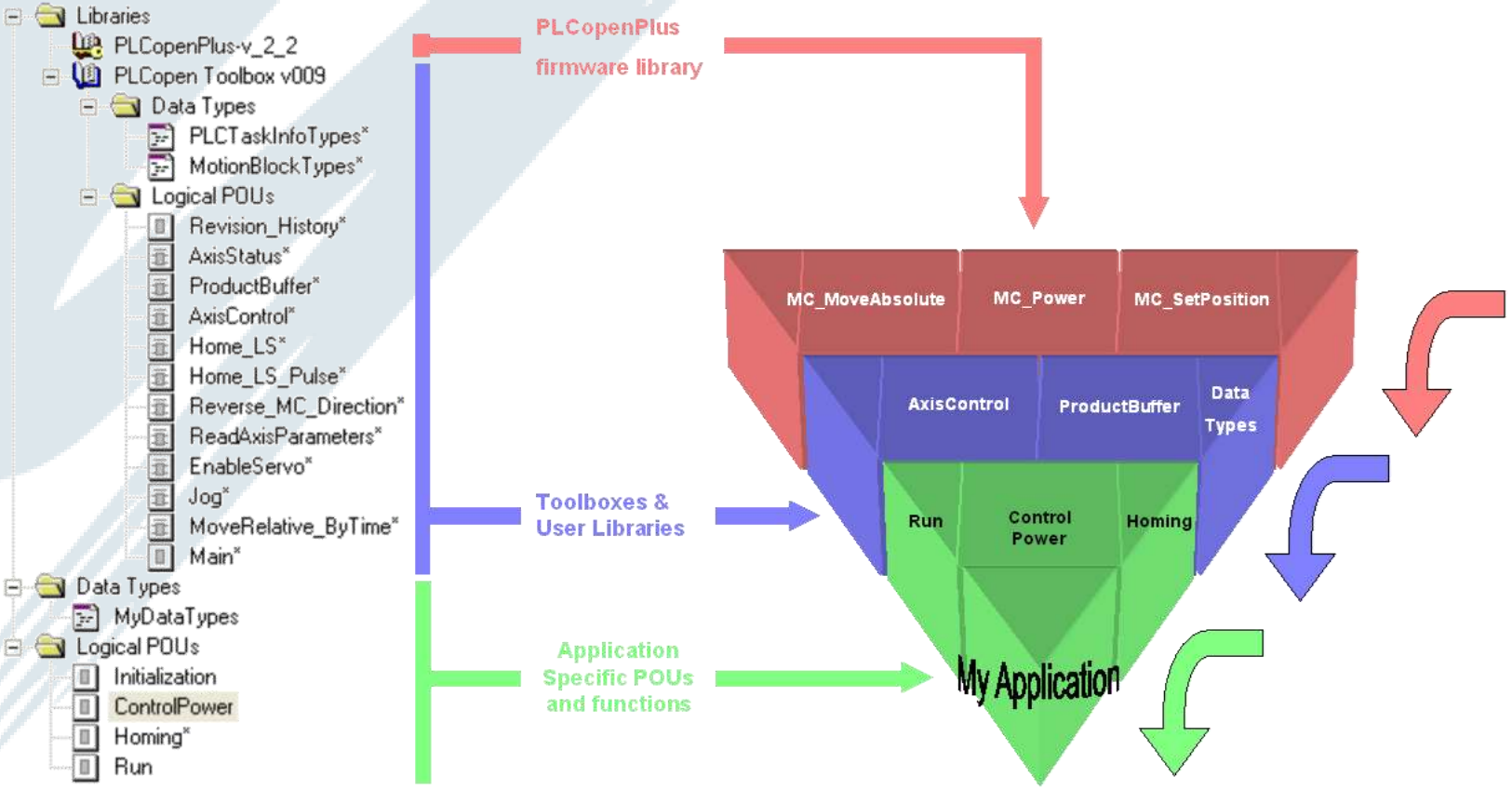
Global Standard

- *IEC = International Electrotechnical Commission*
  - *World standards organization*
  - *Founded in 1906*
  - *International Electrical and Electronic Standards*
  - *50+ Participating countries*

Global Standards



- *Yaskawa Tech Note: TN.MCD.08.130*



All Functions, Function Blocks, and DataTypes from the libraries are available for the application.

5 Languages

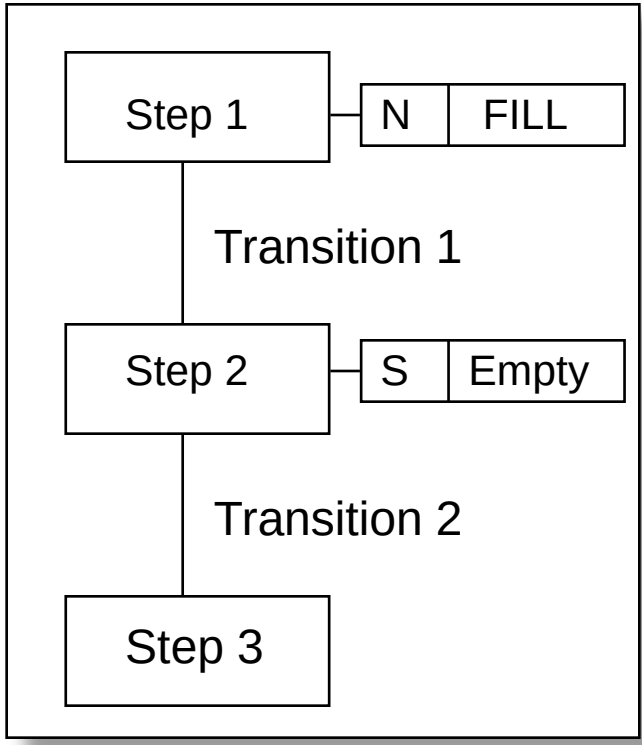
Instruction List (IL)

```
LD      A
ANDN    B
ST      C
```

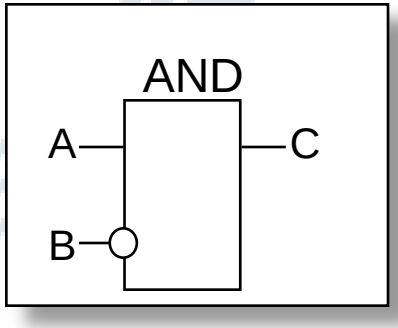
Structured Text (ST)

```
C:= A   AND  NOT B
```

Sequential Function Chart (SFC)



Function Block Diagram (FBD)

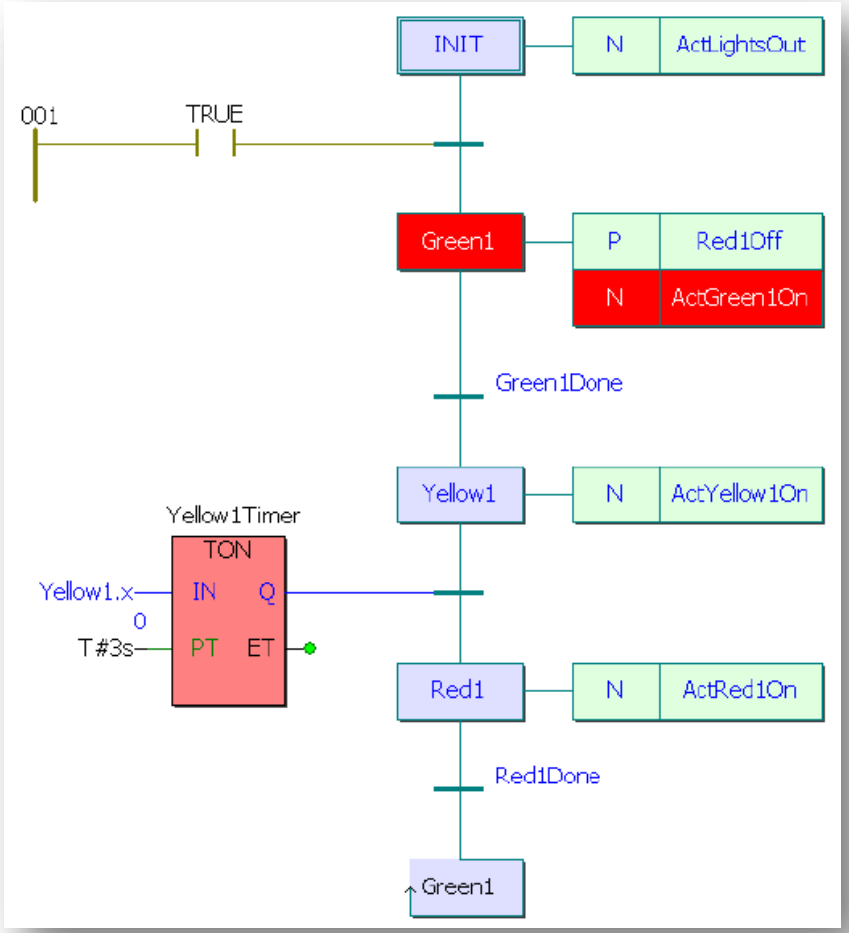


Ladder Diagram (LD)

```
A   B                  C
-| |--|/|----------------(  )
```

What is SFC

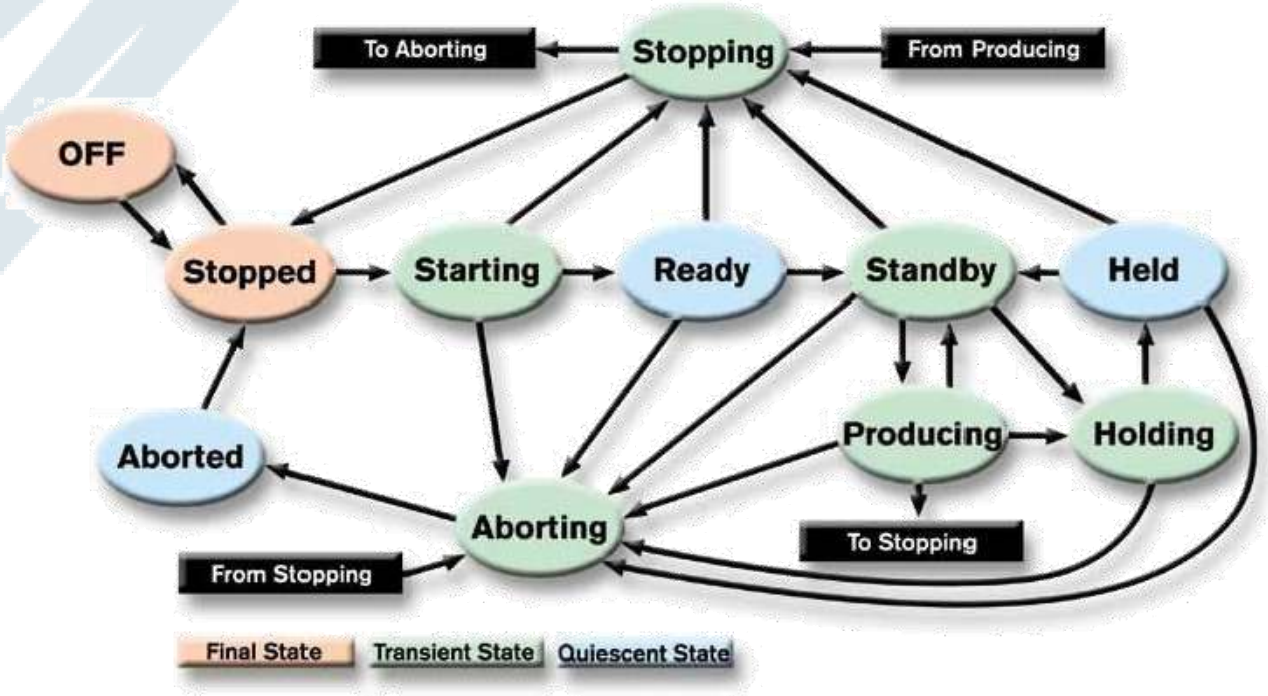
- *A graphical programming language to describe control sequences in graphical form*
- *A tool for "top-down" analysis and representation of a control sequence*
- *Shows the main states of a program*



Traffic Light Example

When to use SFC

- *When there is a definite sequence for the machine to follow*
- *When it is advantageous to think of the machine operation as different states of operation*
- *When someone who is not familiar with your code will have to work with it. Look at SFC and quickly understand how the machine works*



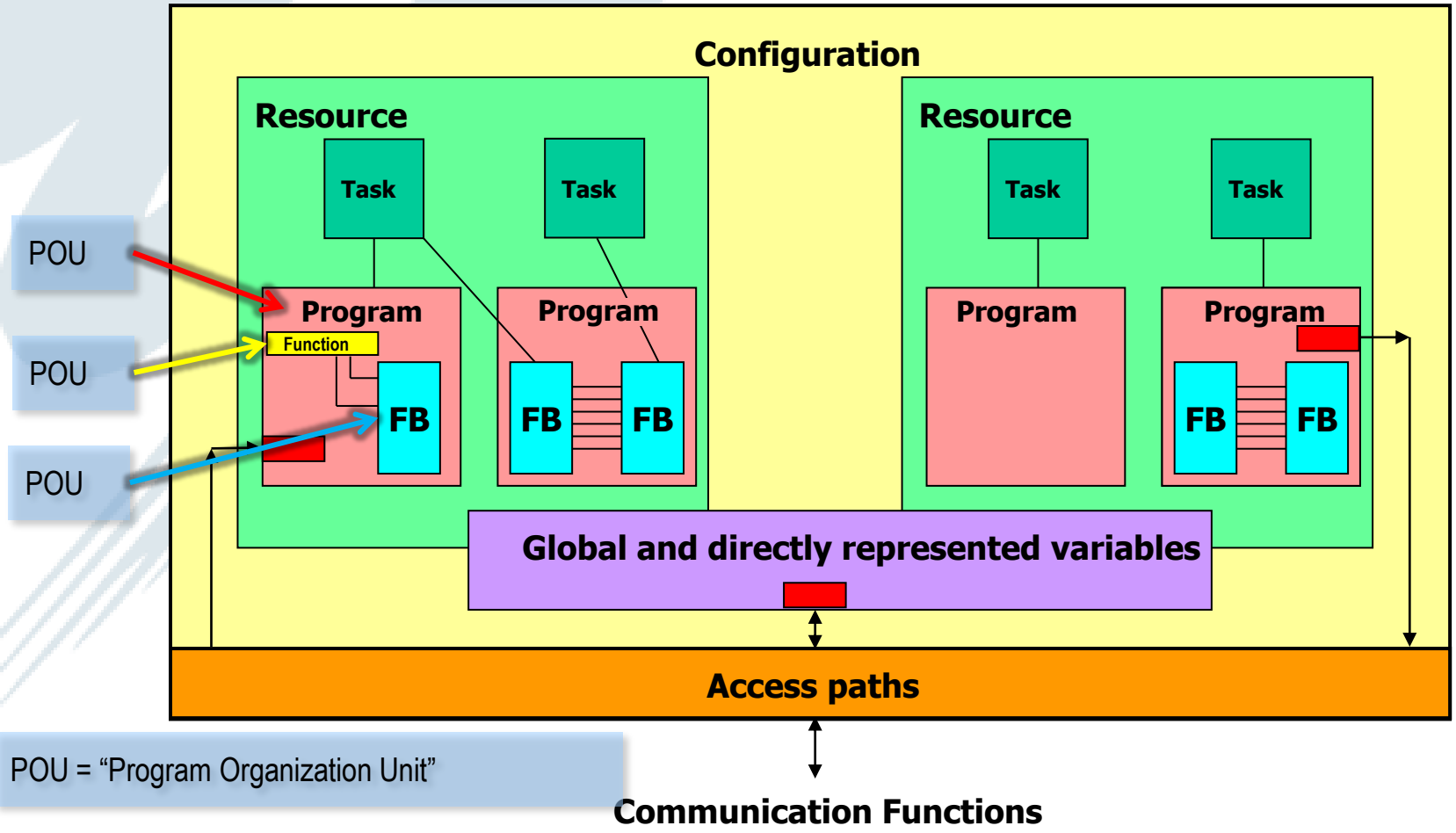
Elementary Data Types

- *Data Types standard with IEC 61131-3*

No.	Keyword	Data Type	Bits
1	BOOL Use	Boolean	1
2	SINT	Short integer	8
3	INT Use	Integer	16
4	DINT	Double integer	32
5	LINT	Long integer	64
6	USINT	Unsigned short integer	8
7	UINT Use	Unsigned integer	16
8	UDINT	Unsigned double integer	32
9	ULINT	Unsigned long integer	64
10	REAL	Real numbers	32
11	LREAL Use	Long reals	64
12	TIME Use	Duration	32
13	DATE	Date (only)	16
14	TIME_OF_DAY or TOD	Time of day (only)	32
15	DATE_AND_TIME or DT	Date and time of day	64
16	STRING	Character string	8
17	BYTE Use	Bit string of length 8	8
18	WORD Use	Bit string of length 16	16
19	DWORD	Bit string of length 32	32
20	LWORD	Bit string of length 64	64

Software Model

- *IEC hardware must operate according to this standard*



POU = “Program Organization Unit”

Configuration

MotionWorks IEC does not yet support an entire configuration with multiple controller resources

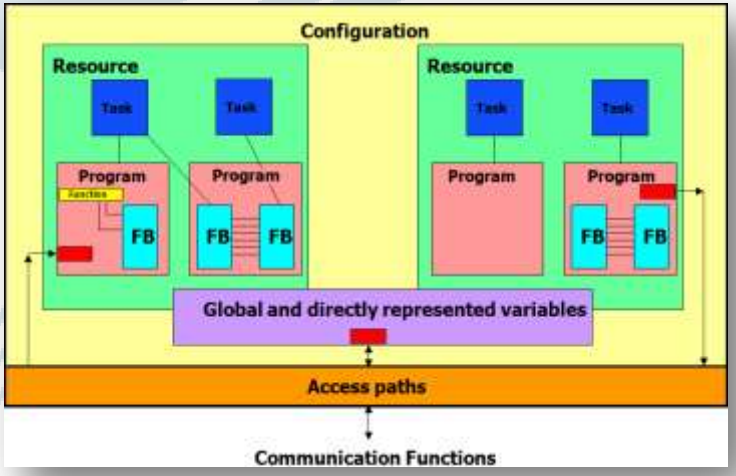
Software Model

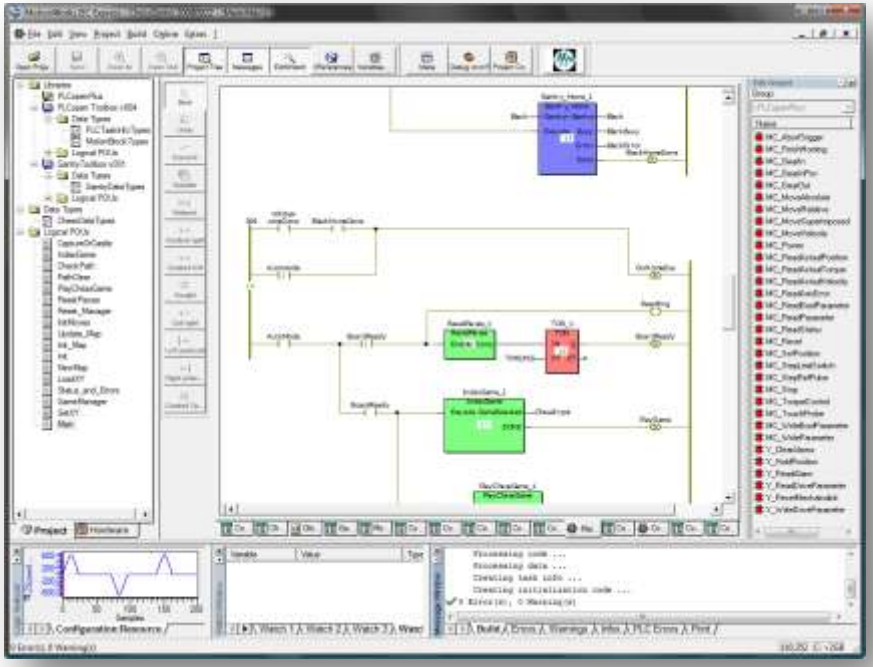
Configuration

Resource

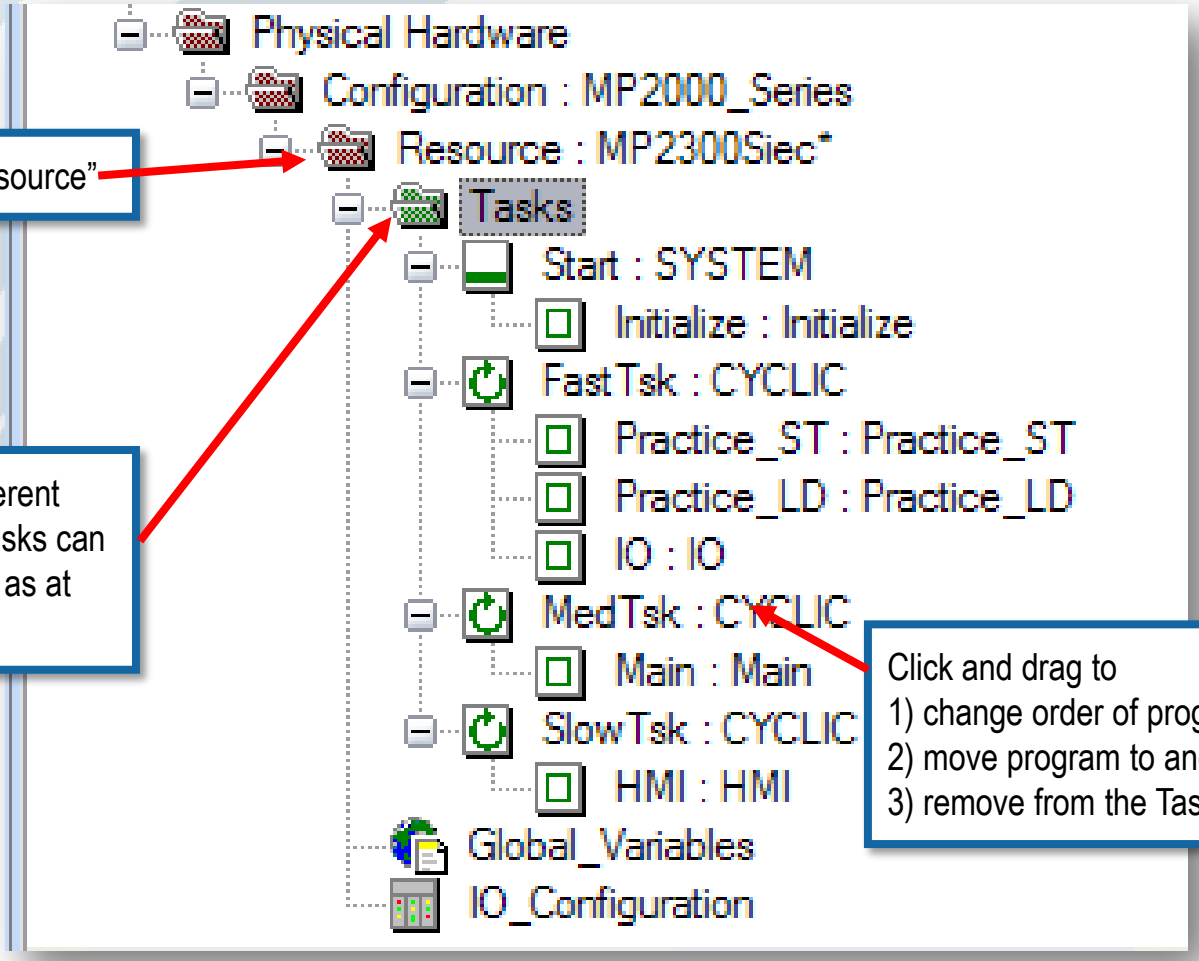
Resource, Task, POU

- *Controller "Resource" must execute code according to IEC standards*





- *Task Details*



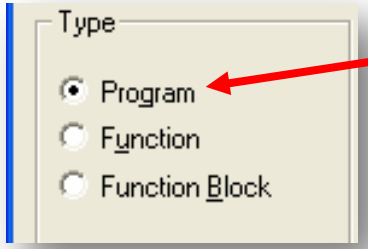
The controller is the "Resource"

Multiple tasks run at different update rates. System tasks can be set to run once, such as at WarmStart

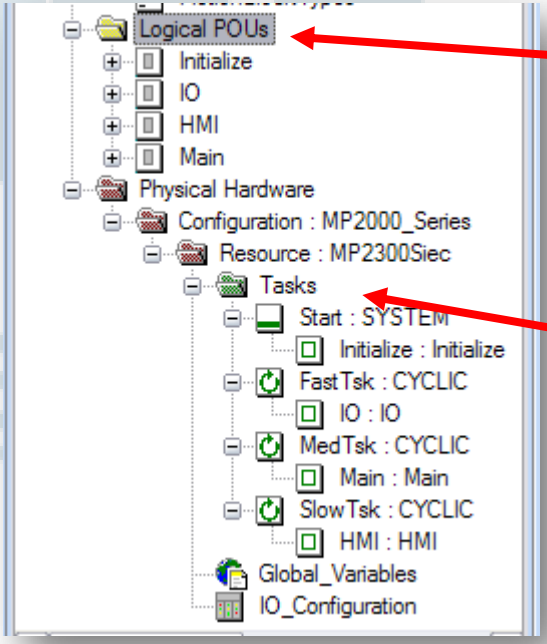
Click and drag to
1) change order of program execution
2) move program to another task
3) remove from the Task list

POU Execution

- *Program Organization Unit (POU)*
  - *3 types of POU*
    - » *Program*
    - » *Function*
    - » *Function Block*

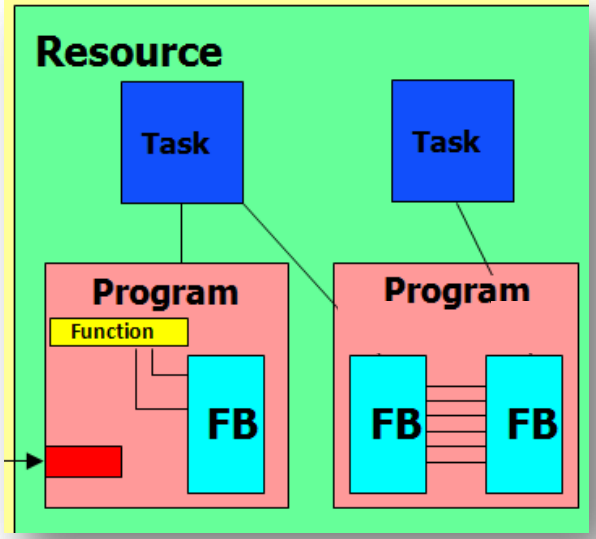


Only Program POUs can run in a task. Functions and Function Blocks run in a program.



Logical POUs contain the code pages

Physical Hardware shows where the **Program POUs** run



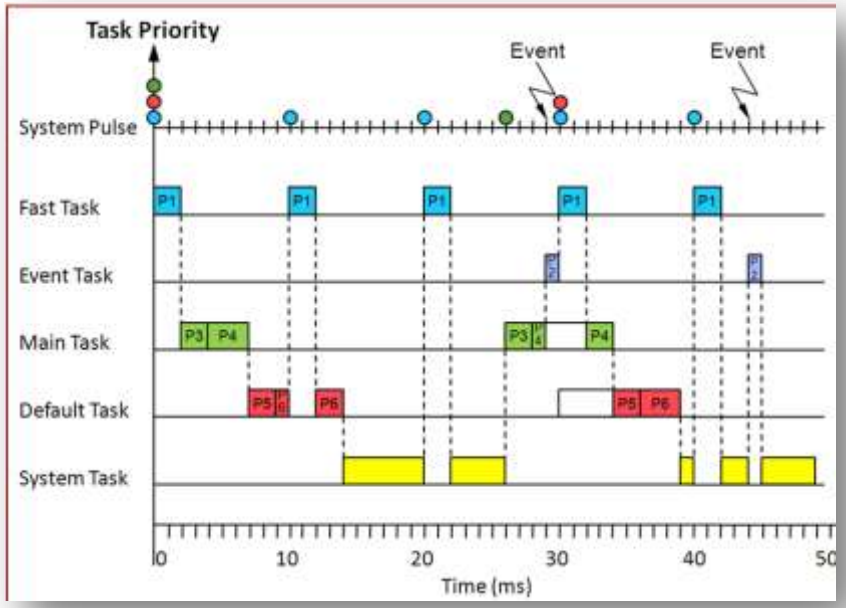
Multi Task

New Task
Task Types
System Task, Default Task
Cyclic Task
Slow Task
Task Priority
Watchdog
Task Properties/Settings

YASKAWA

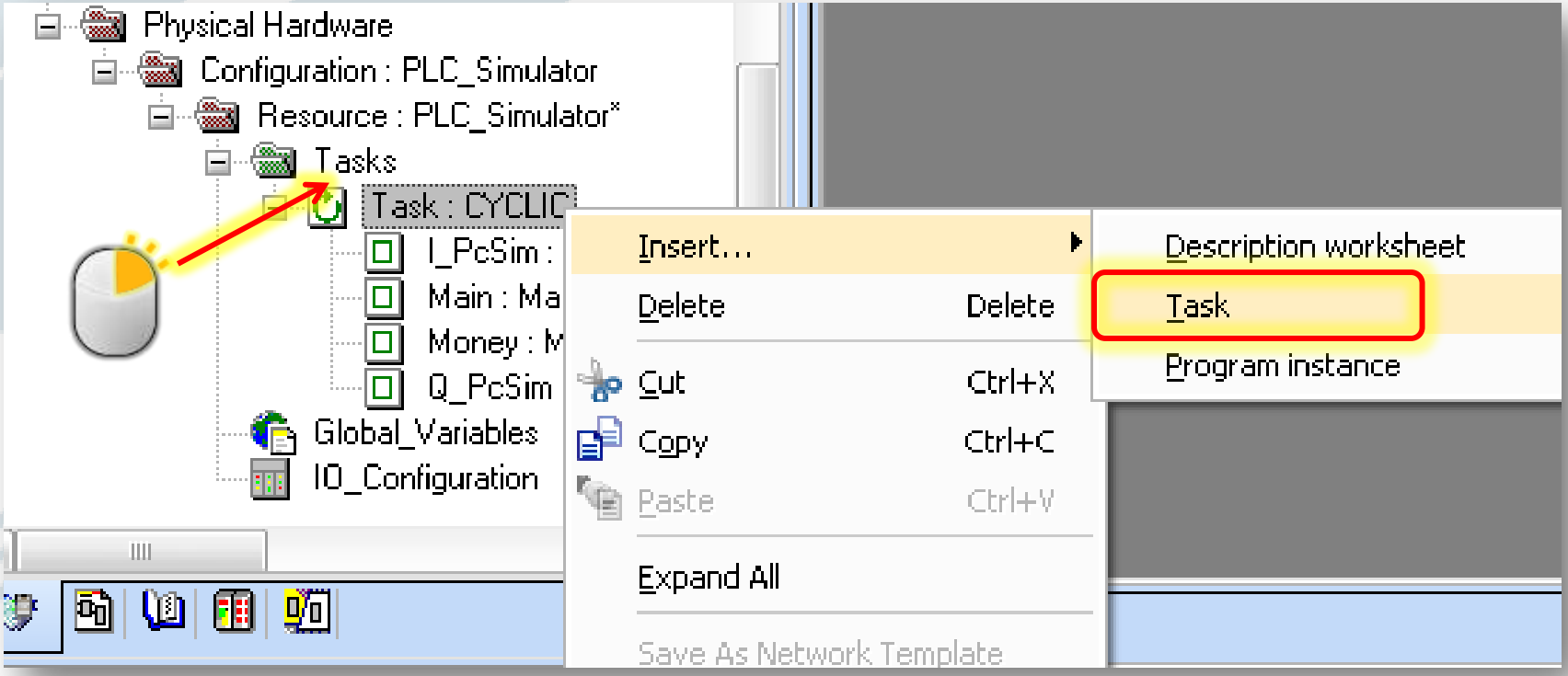
Multi-Task Overview

- *4 Task Types*
  - *System, cyclic, default, event*
- *Up to 16 Tasks*
  - *Yaskawa implementation*
- *Task Interval for Cyclic Tasks*
  - *Lowest interval depends on PLC*
    - » *Simulator ~ 10 ms*
    - » *MP2000iec ~ 1 ms*
    - » *MP3000iec ~ 0.125 ms*
- *Task Priority for Cyclic Tasks*
  - *0 = highest priority (runs first)*
  - *15 = lowest priority (runs last)*
- *Task Execution*
  - *Read Inputs %I*
  - *Program POUs*
  - *Write outputs %Q*



Insert New Task

- *Class Project*
- *Right-Click Task Folder*
- *Insert → Task*



Task Types

Default

- *Executes after all other tasks complete*

Cyclic

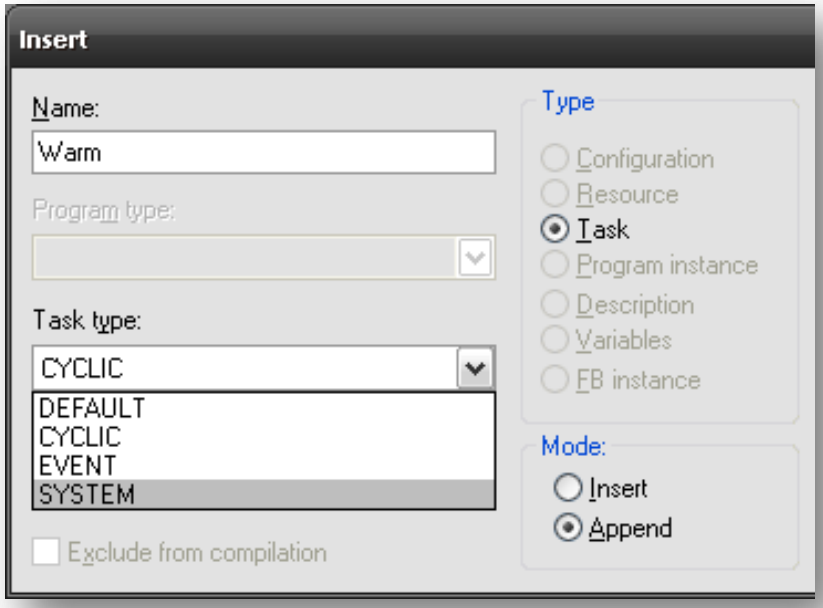
- *Repeats at given “interval”.*
  - » *Slow interval for non-critical tasks.*
  - » *Fast interval for machine IO and motion*
- *Priority*
  - » *Which cyclic task is done 1st, 2nd, 3rd, etc*

Event

- *Executes 1 time when condition is met*

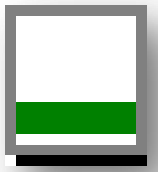
System

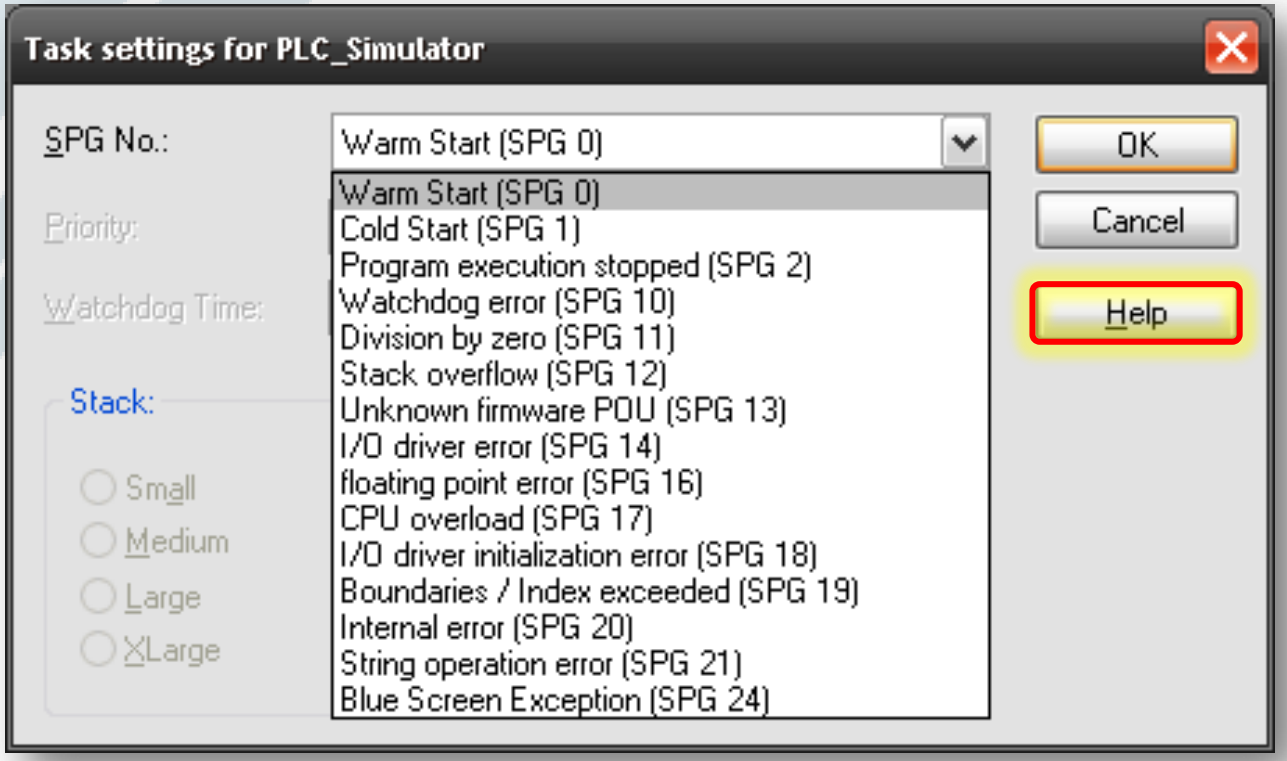
- *Executes 1 time when the system reaches a given state*
  - » *Warm Start*
  - » *Cold Start*



System Tasks

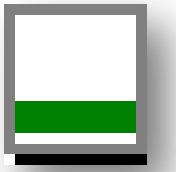
- *Use system tasks to execute a program POU triggered by specific system conditions*
  - *Warm Start*
  - *Cold Start*
  - *Errors*

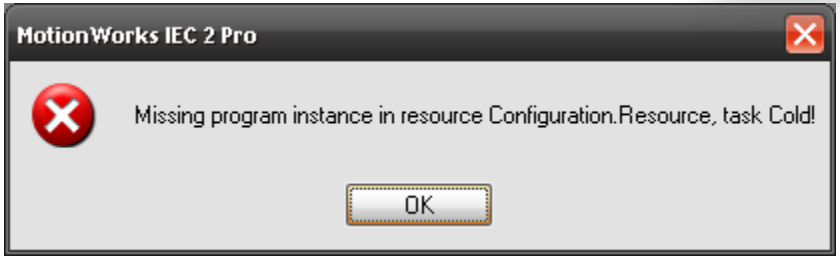




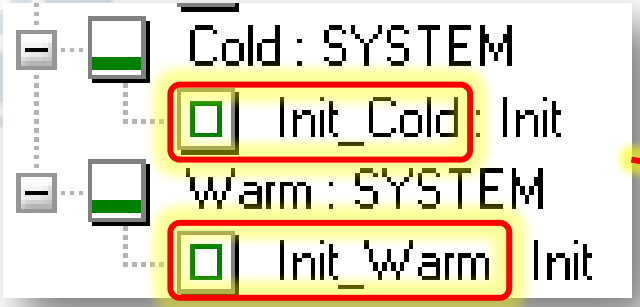
Create System Tasks

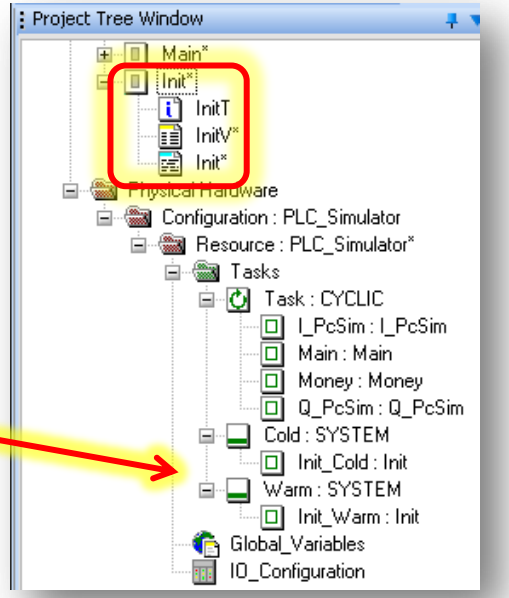
- *Create a warm start system task named “warm”*
- *Create a cold start system task named “cold”*





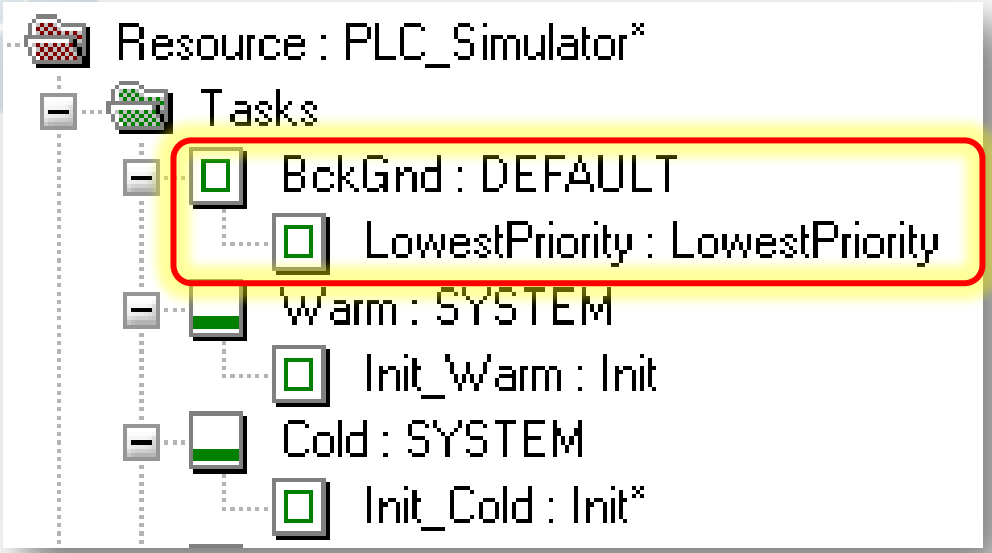
- *Create a blank ST POU named “Init”*
  - *Insert program instance to Cold task*
  - *Insert program instance to Warm task*





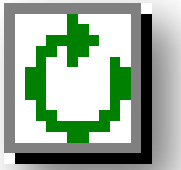
Create Default Task

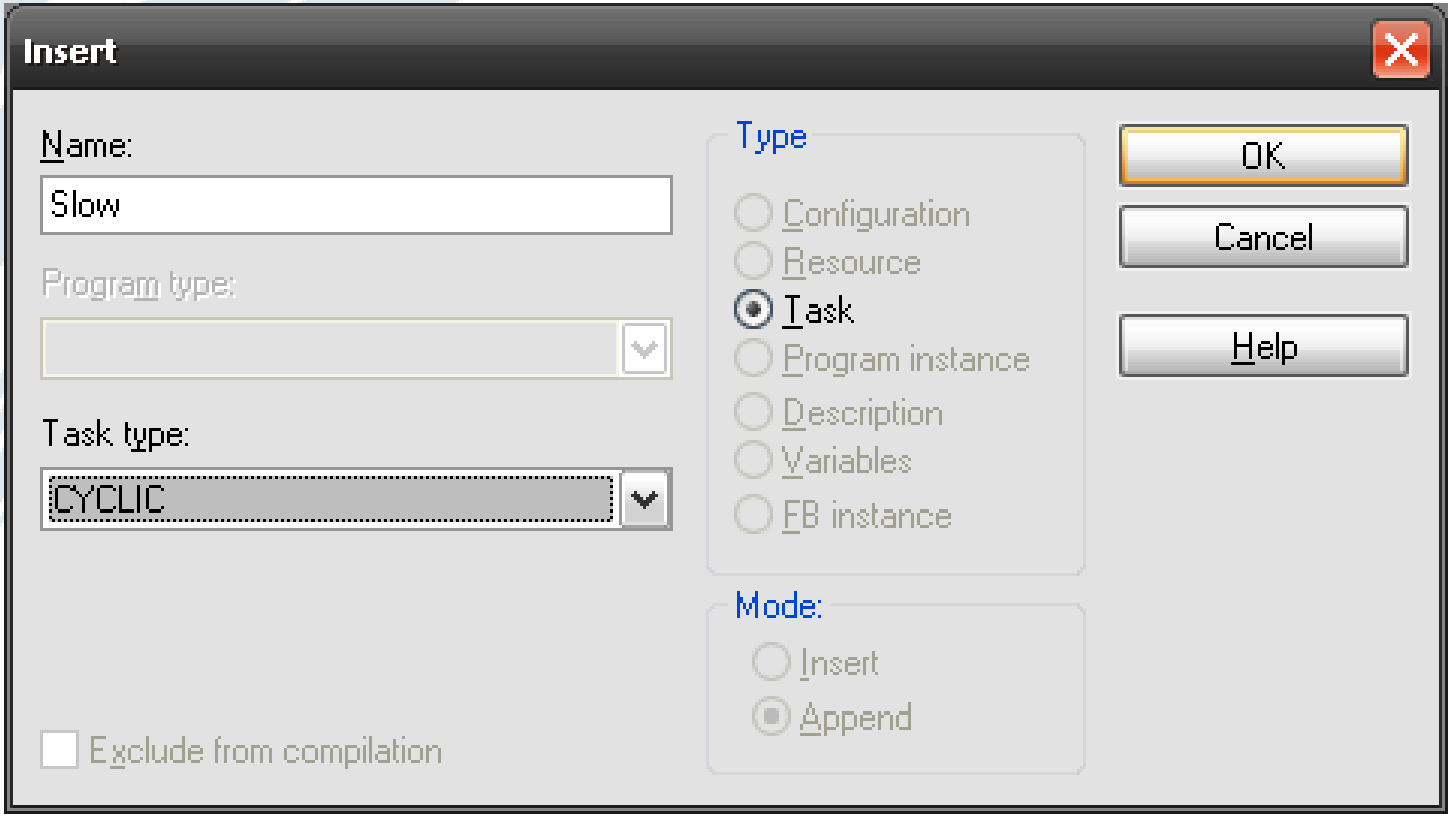
- *Create a Default task named “BckGnd”*
  - *500ms watchdog*
- *Create a blank ST program POU named “LowestPriority”*
  - *Insert program instance in BckGnd task*



Create Cyclic Task

- *Create a cyclic task named “Slow”*
- *Why have another cyclic task?*
  - *Optimize code execution*
  - *Critical POUs in fast tasks (ex: motion & machine IO)*
  - *Non-time-critical POUs in slow tasks (ex: user interface)*

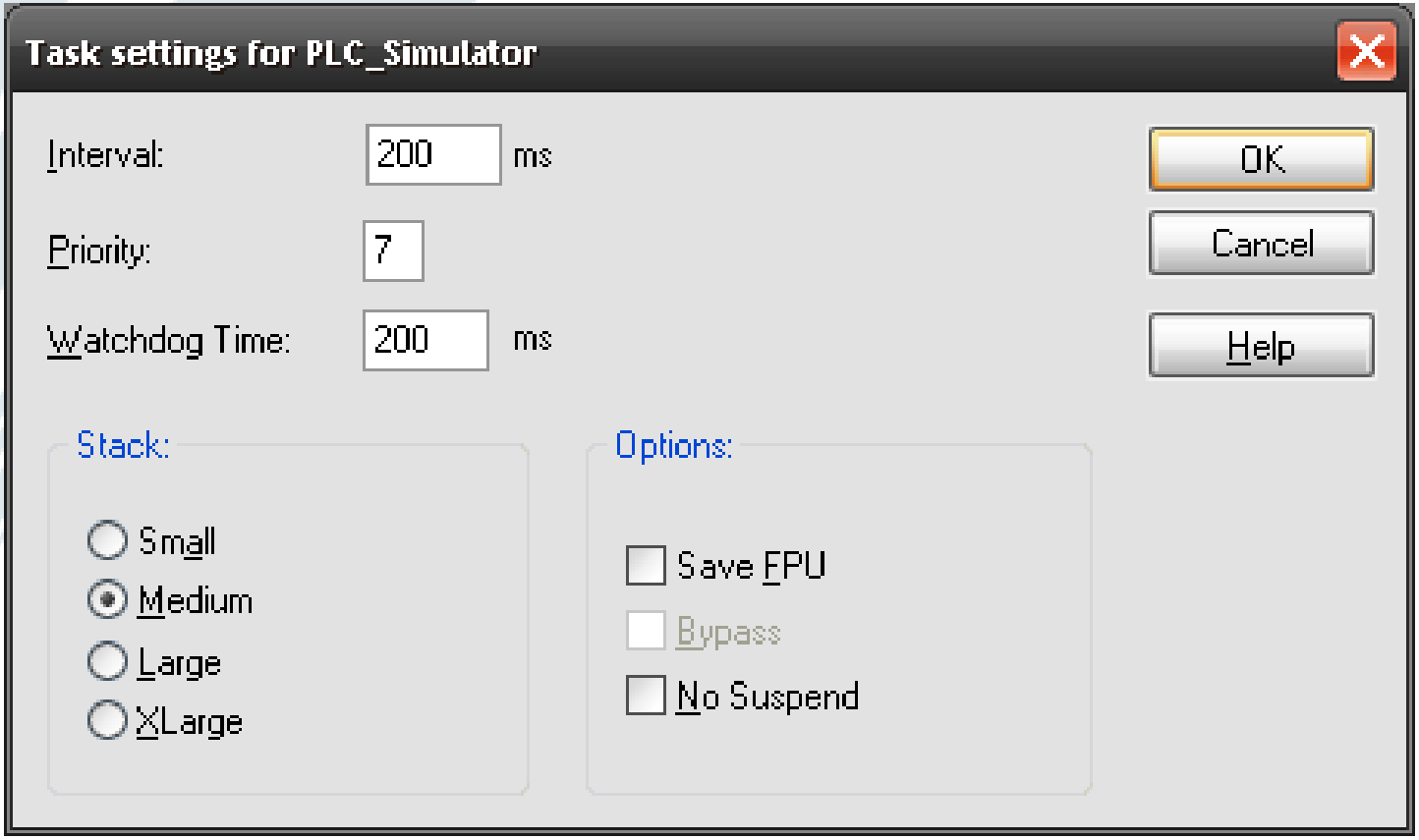




Slow Task Settings

- *Set interval time = 200 ms*
- *Watchdog Time = Interval*
- *Priority = 7*

Why Priority 7?
To leave room for higher priority tasks in the future ("medium" etc)



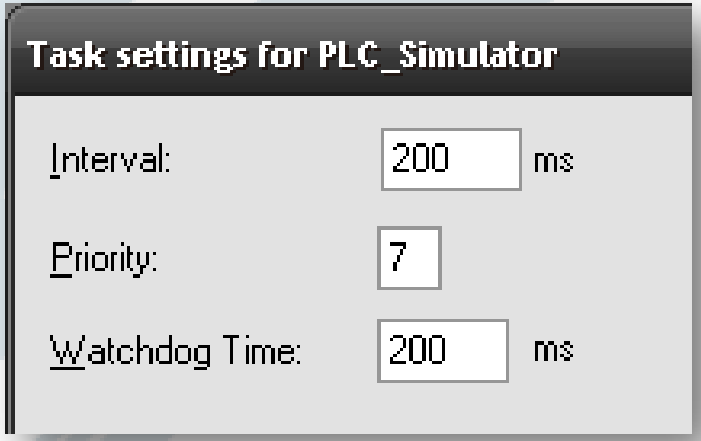
Task Priority

- *Tasks are executed sequentially in order of priority*
  - *0 = first, 15 = last*
    - » *Software does allow entry up to 31, but don't do it! Controller internally executes hidden tasks beyond priority 15. You want your code to execute before these, so keep it to 15 and below*
- *Similar to prioritizing office work*
  1. *Phone Calls*
  2. *Emails*
  3. *Daily Report*
  4. *Weekly Report*
  5. *Monthly Report*
  6. *Yearly Goals*

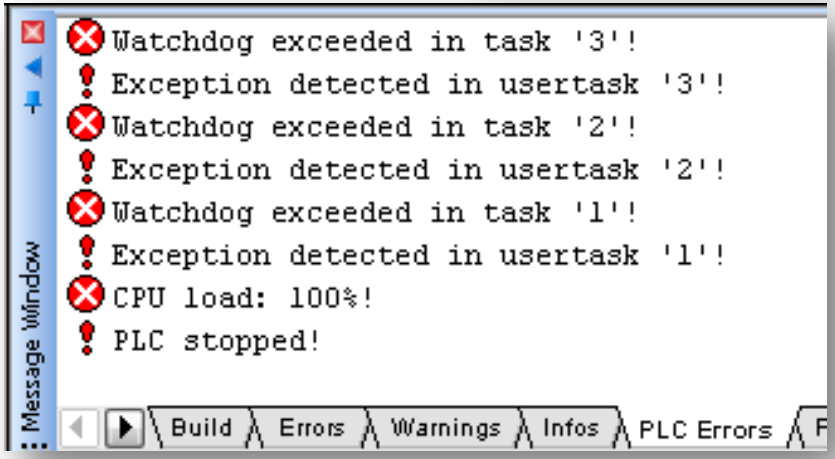
You wouldn't work first on the yearly goal while refusing to answer phone calls and emails!

Watchdog Time

- *Maximum time allowed for task to complete*
  - *Use same setting as Interval*

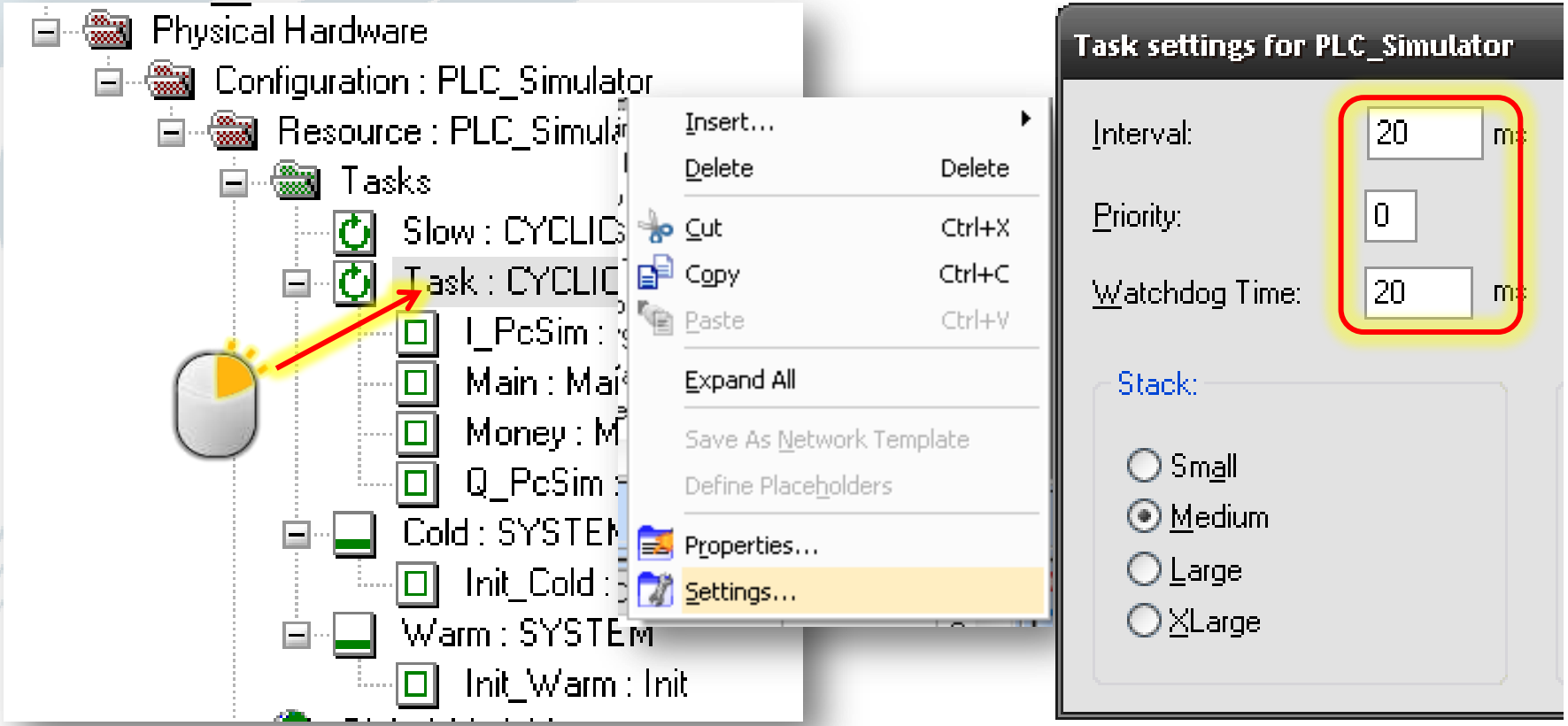


If the task does not complete in time, the PLC will stop with Watchdog error



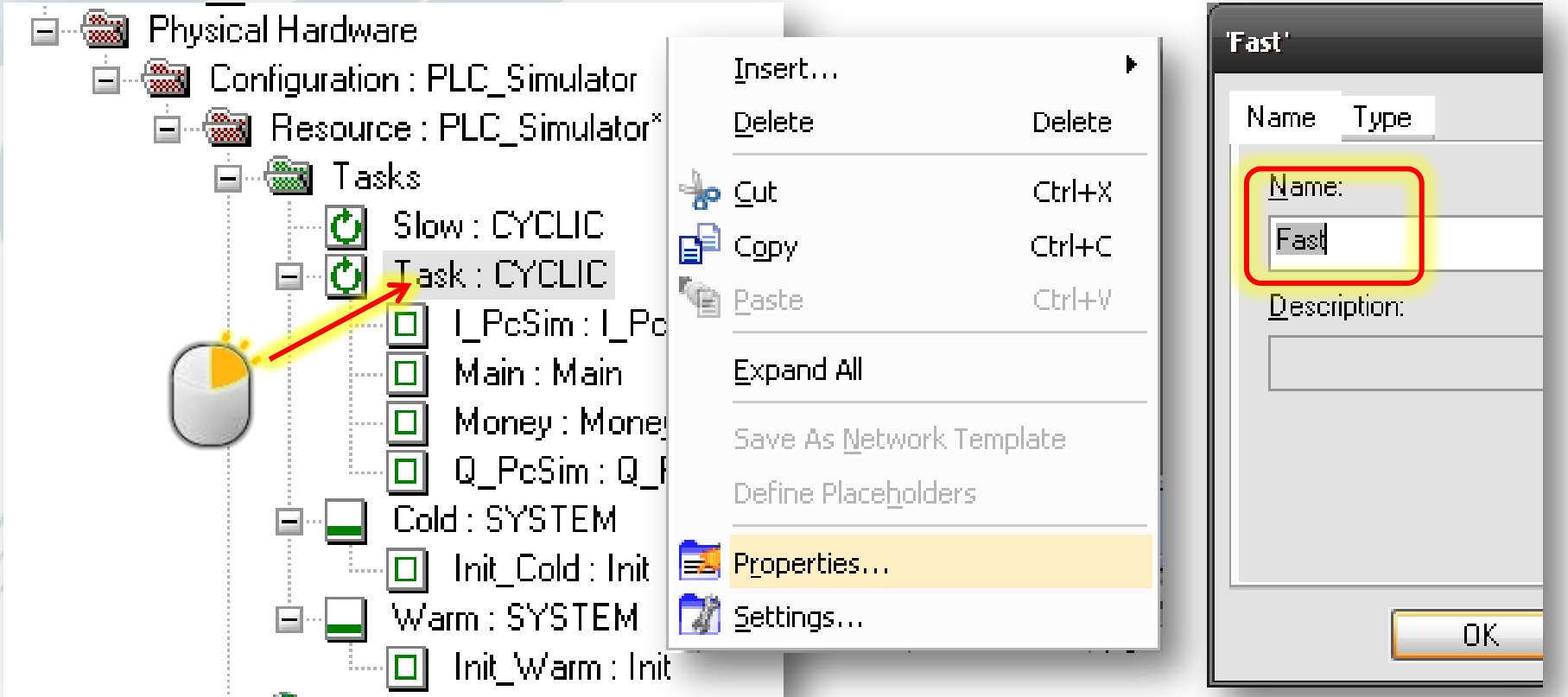
Change Task Settings

- *What are the settings of the existing "task"?*
  - *Right click, settings*
  - *Change to 20 ms*



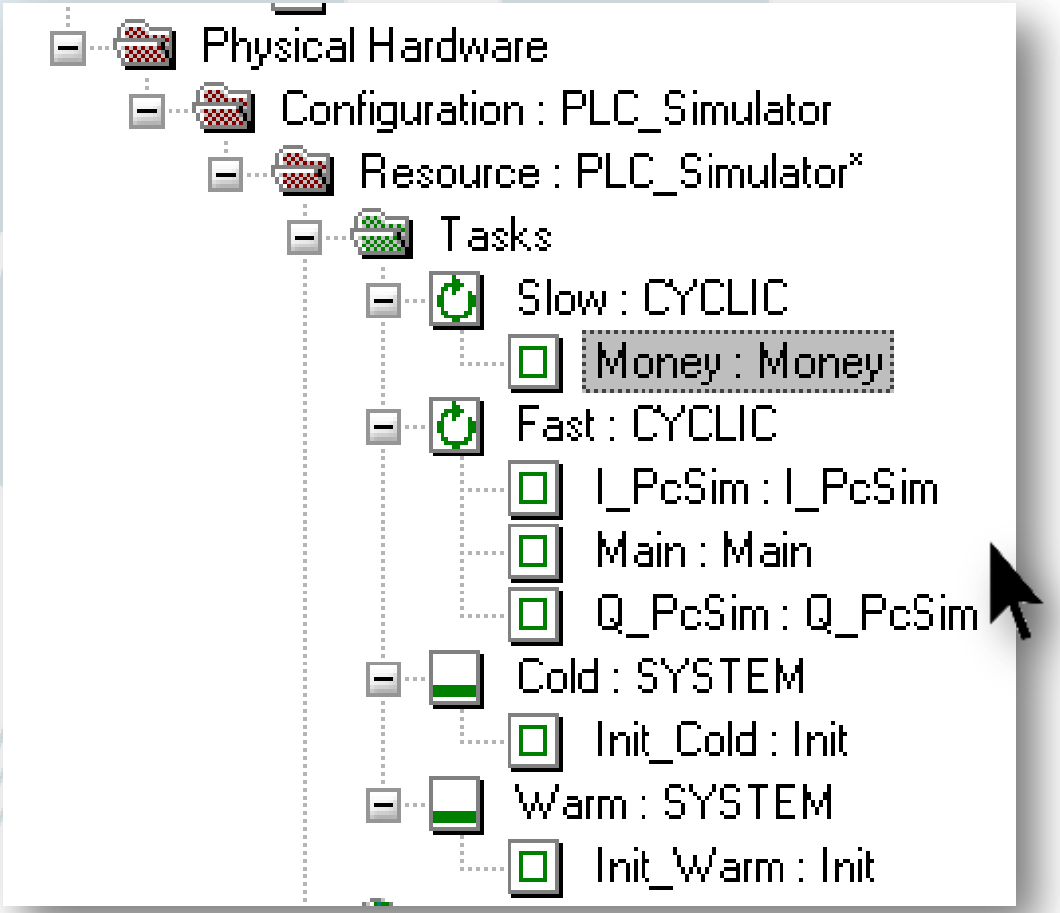
Change Task Properties

- *Change name to “Fast”*



Move Program Instance

- *Click and drag to move*



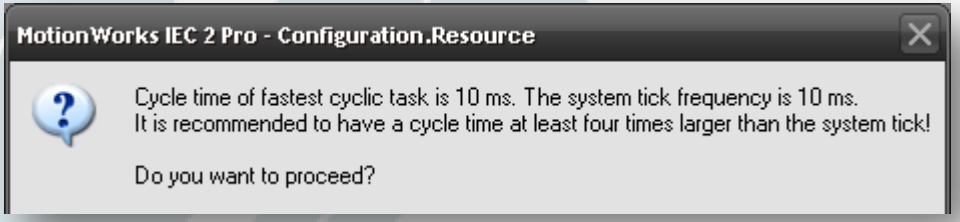
What is the Slow task cycle?

What is the Fast task cycle?

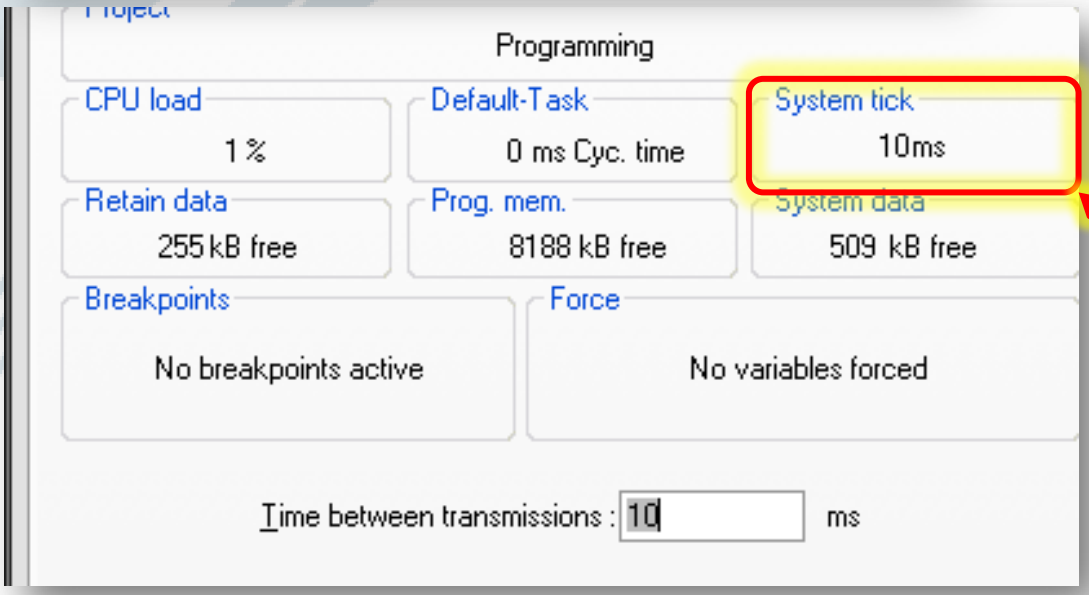
Which POU can be executed at slower cycle?

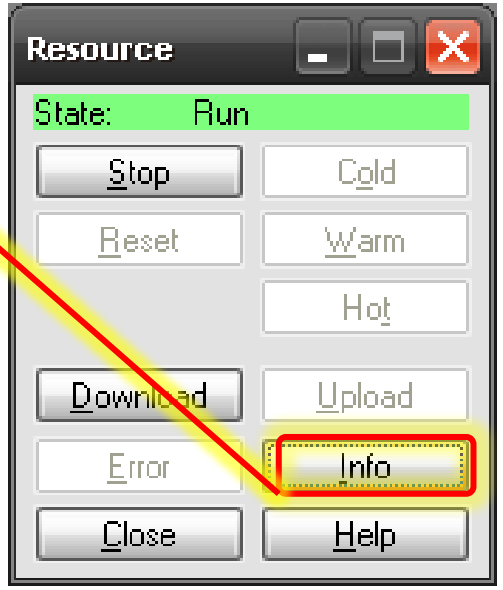
System Tick

- *Fastest possible task interval*
- *Set task interval as integer multiple of system tick*
- *MPiec System Tick = Mechatrolink network update*



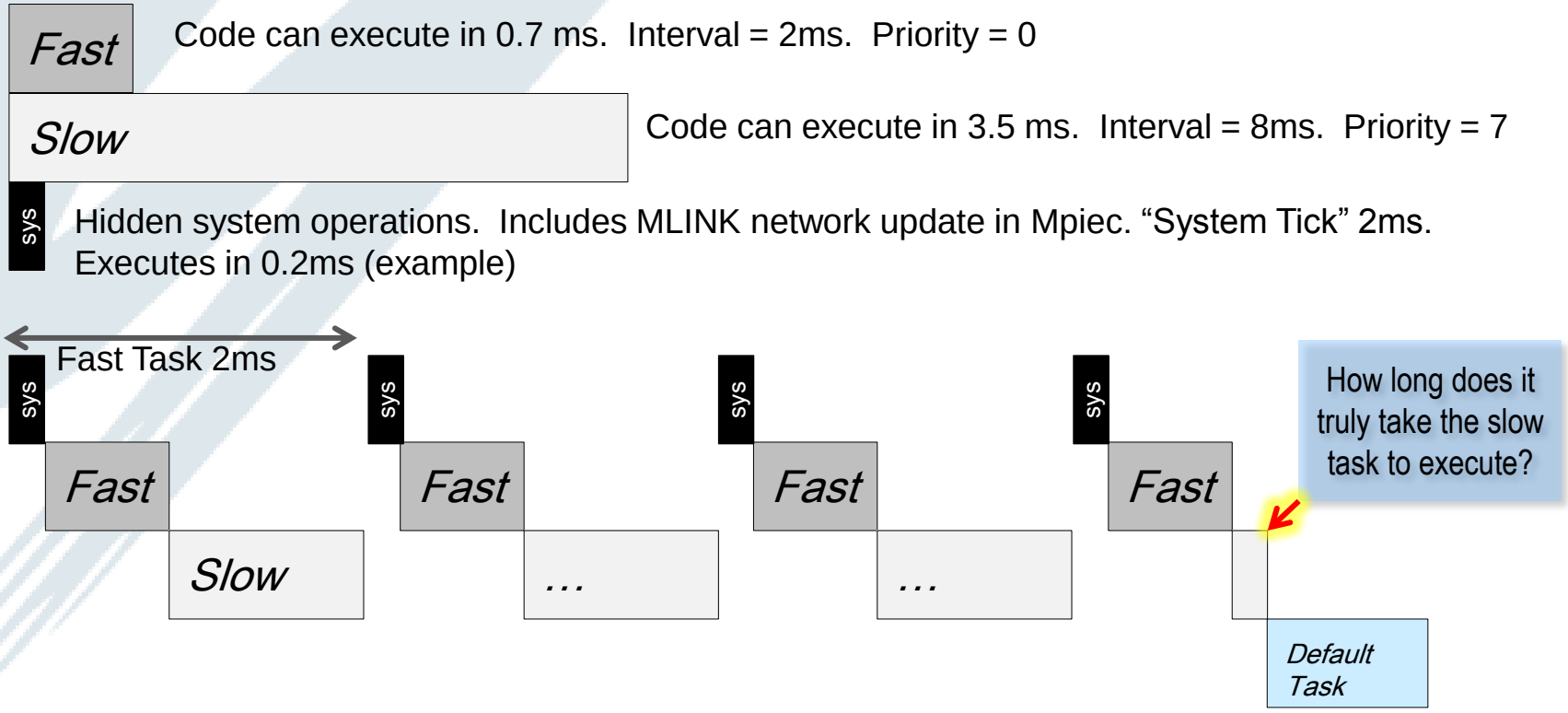
OK to click accept





Interval and Priority

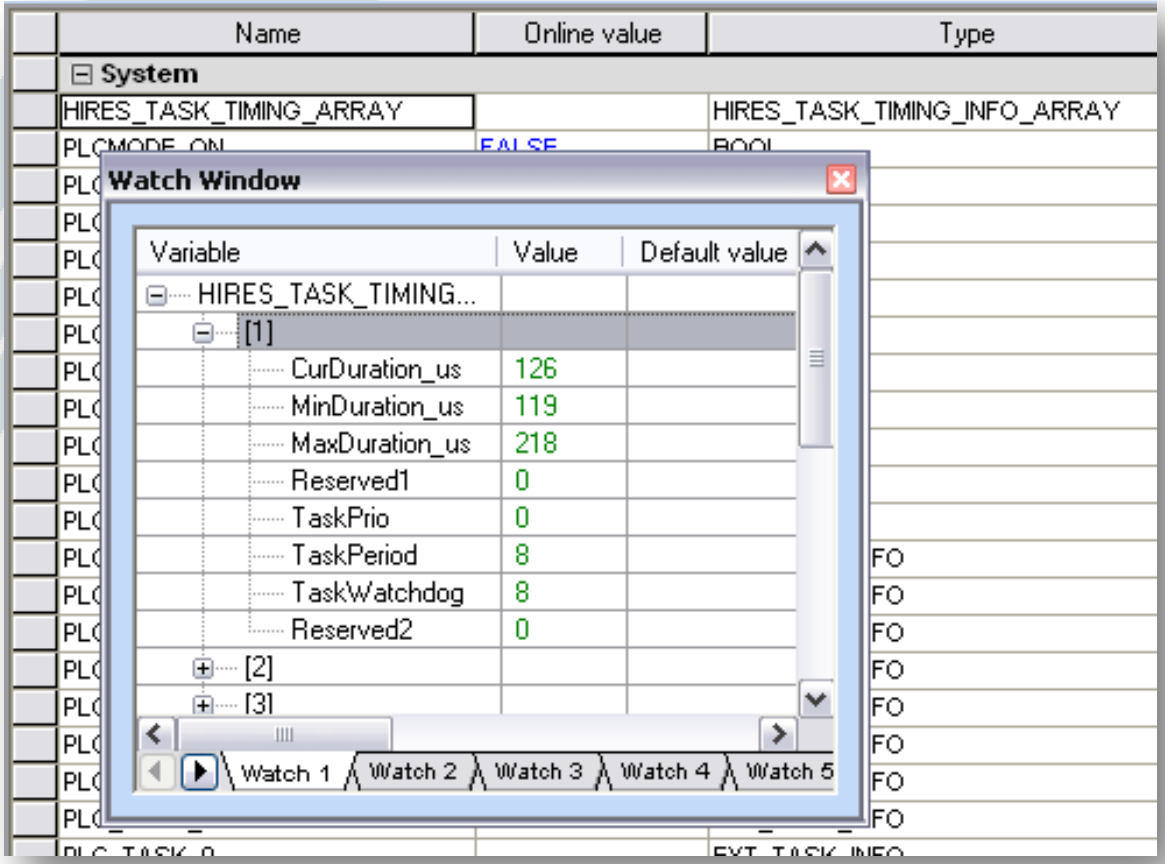
- *One task can interrupt another*
  - *Critical "fast" tasks can interrupt non-critical "slow" tasks*



What important item is missing from the timeline?

Monitor Task Time

- *Supported by MPiec controllers (not by simulator)*
  - *HIRES_TASK_TIMING_INFO_ARRAY*



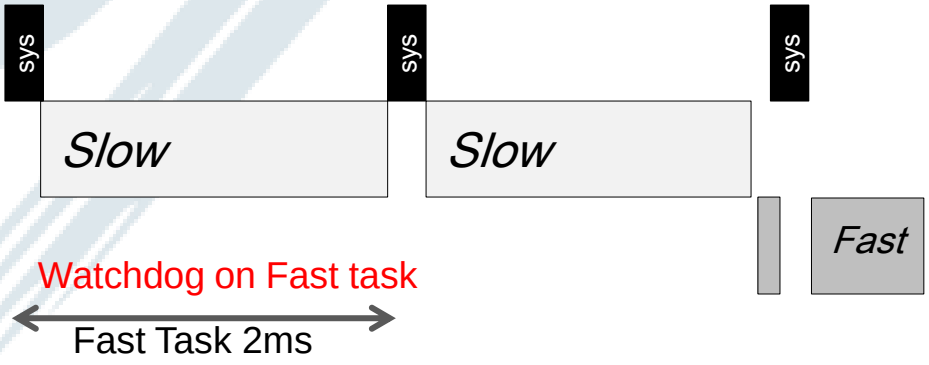
Interval and Priority

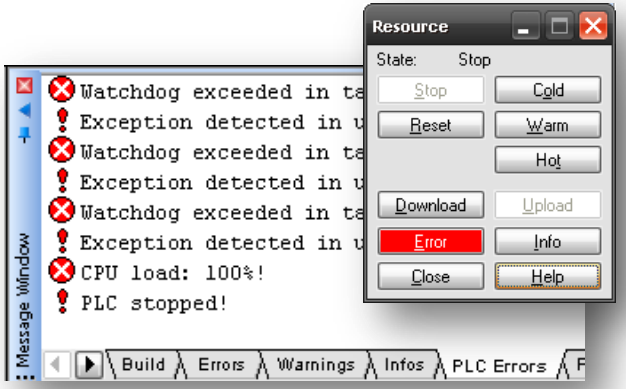
- *What happens with incorrect priority?*

Fast — Code can execute in 0.7 ms. Interval = 2ms. **Priority = 1**

Slow — Code can execute in 3.5 ms. Interval = 8ms. **Priority = 0**

sys — Hidden system operations. Includes MLINK network update in Mpiec. “System Tick” 2ms. Executes in 0.2ms (example)

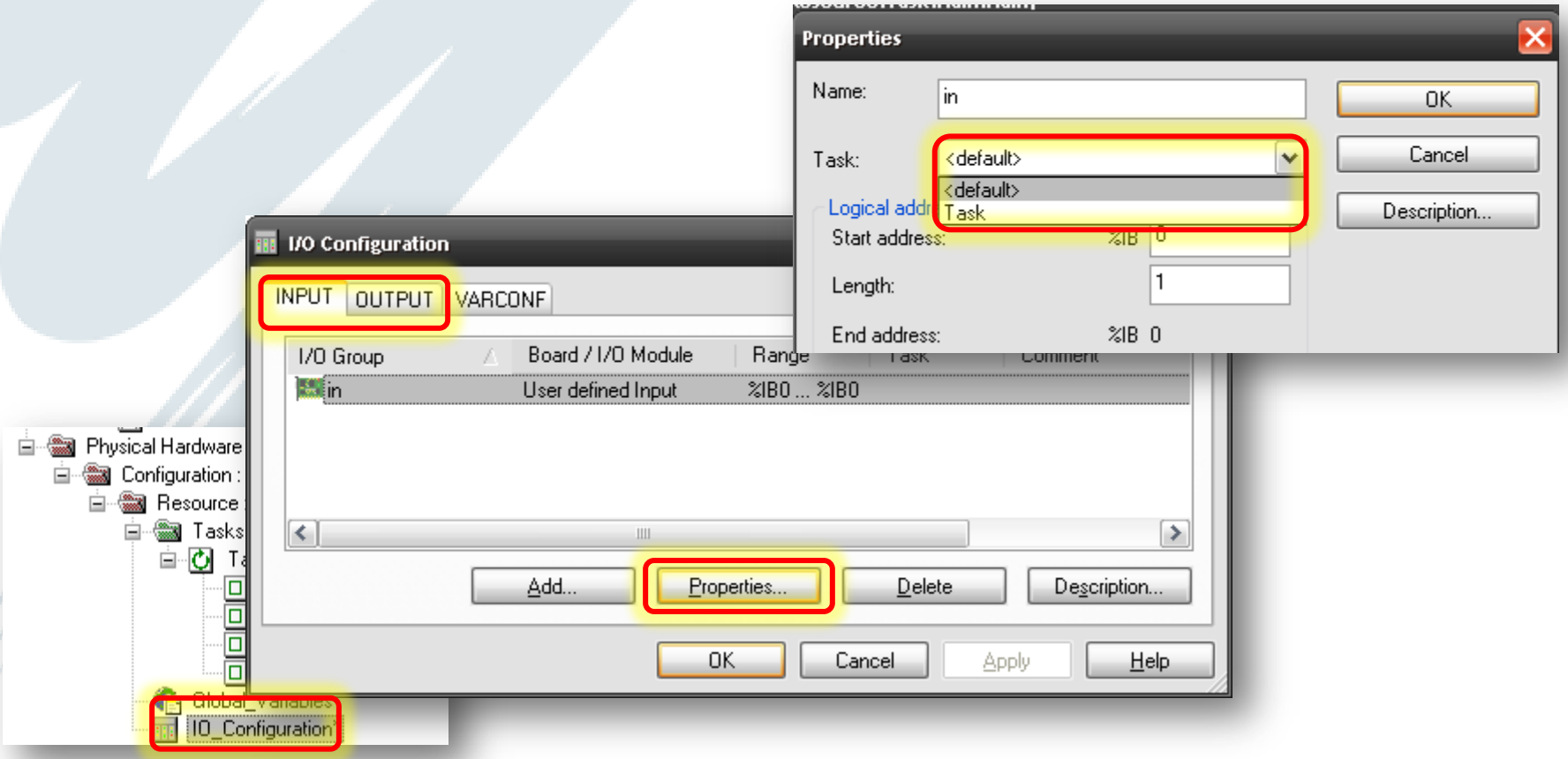




Slow Task 8ms

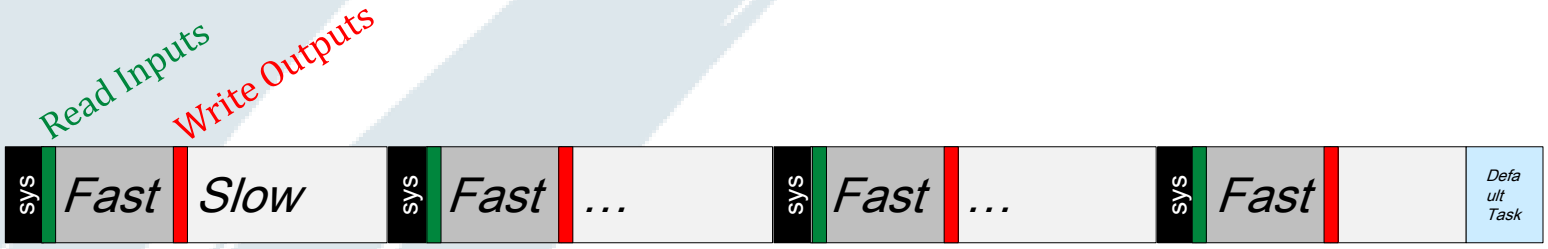
IO Task Assignment

- *Assign IO to the task in which they are used*
  - *Physical Hardware – IO_Configuration: Properties - task*



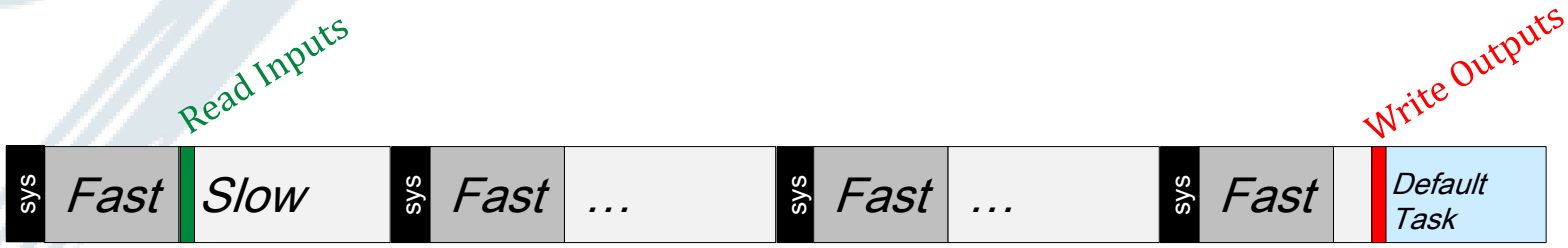
IO Execution Timing

- *IO configuration assigned to **Fast** task*



What if an input changes state rapidly?

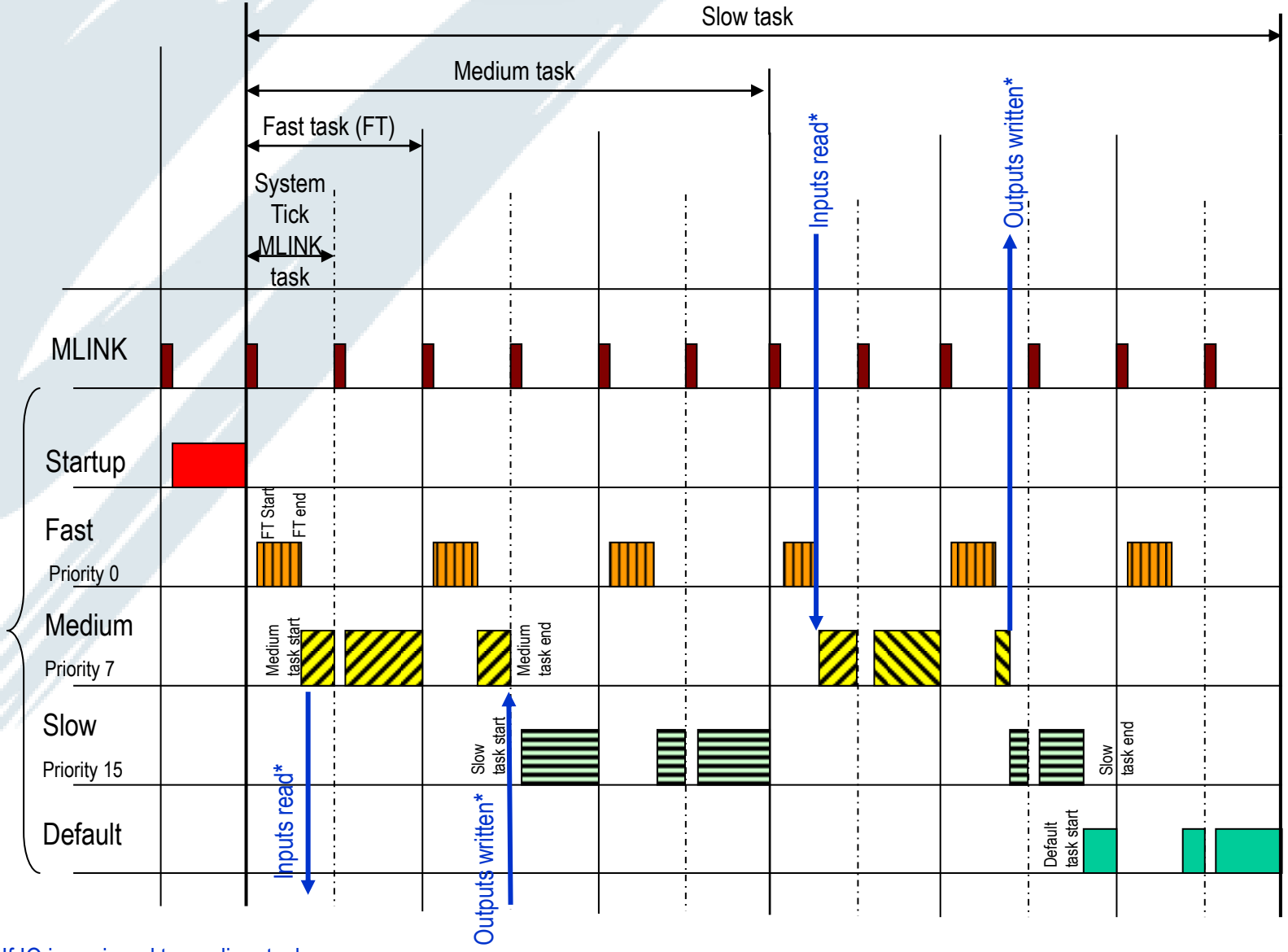
- *IO configuration assigned to **Slow** task*



Inputs not read until task execution begins

IO update takes processor time

Task Execution



* If IO is assigned to medium task

Driving Value

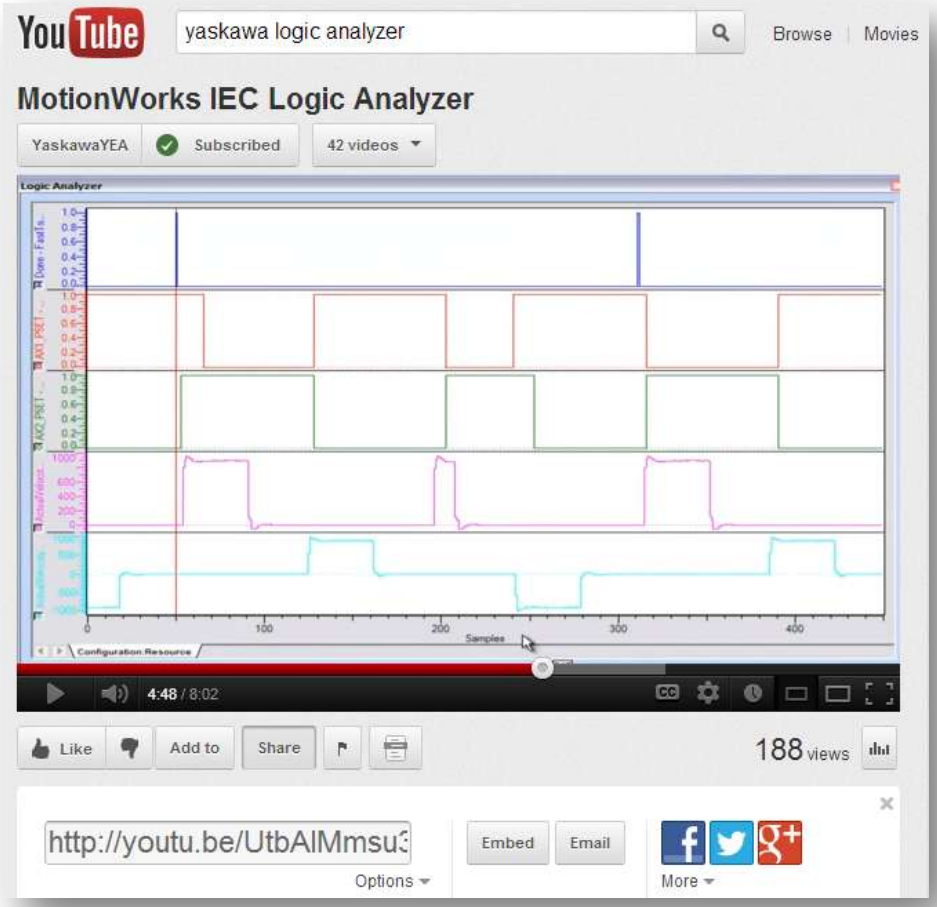
Logic Analyzer

Overview
Add Variables
Trigger Setup
Zoom, Cursors

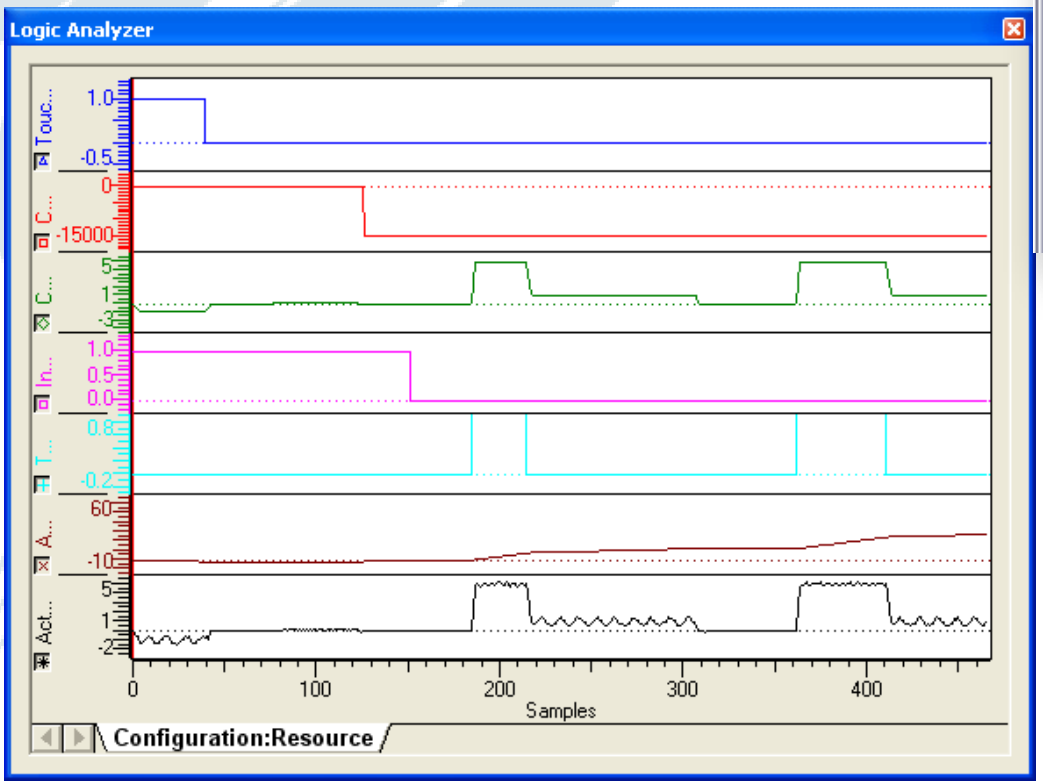
YASKAWA

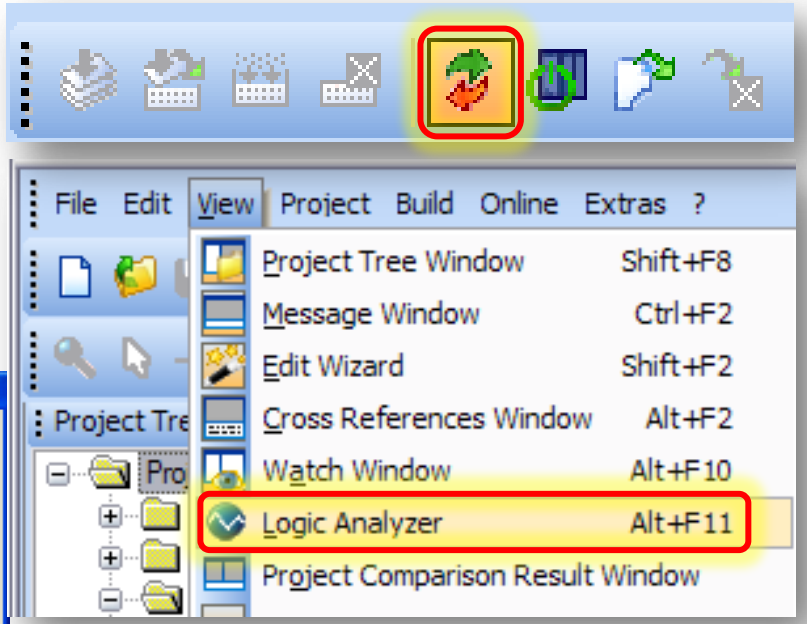
- *Quick Reference Guide*
  - *Document: QRG.MP2000iecSeries.01*
  - *Search Yaskawa.com: QRG*

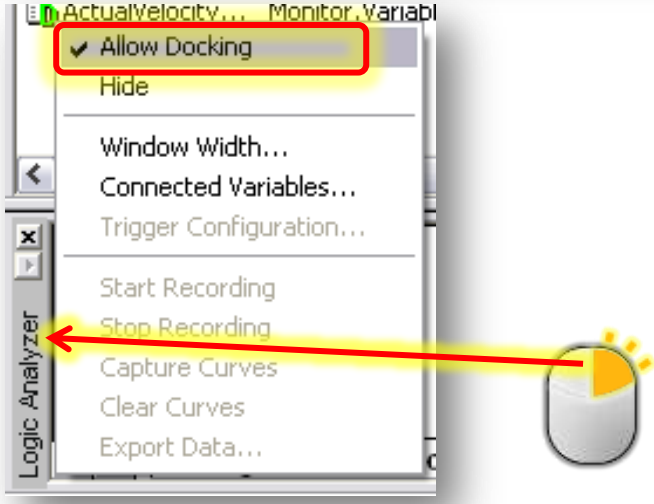
- *eLearning Video on Yaskawa.com*
  - *Document: eLV.MotionWorksIEC.01.LogicAnalyzer*
- *YouTube Video*
  - *Search: Yaskawa Logic Analyzer*



- *Features*
  - *16 variables*
  - *Sample at task update rate*
  - *With MotionWorks IEC open*
  - *Debug mode*

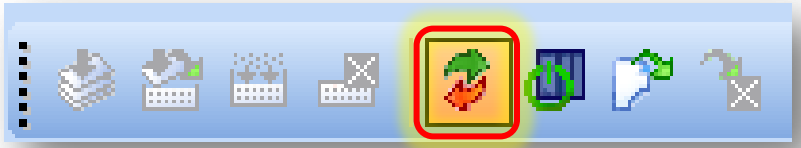


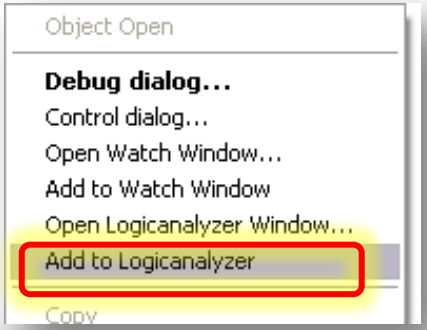




Add Variables

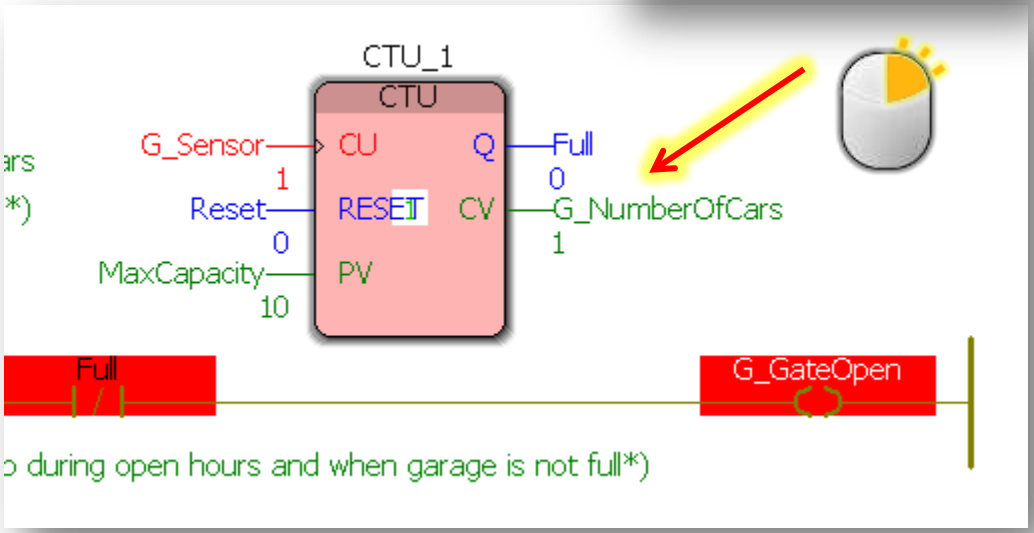
- *Add variables to Logic Analyzer*
  - *Debug ON*
  - *Right-Click Variable*
    - » *From Code*
    - » *From List*
  - *Add to Logicanalyzer*





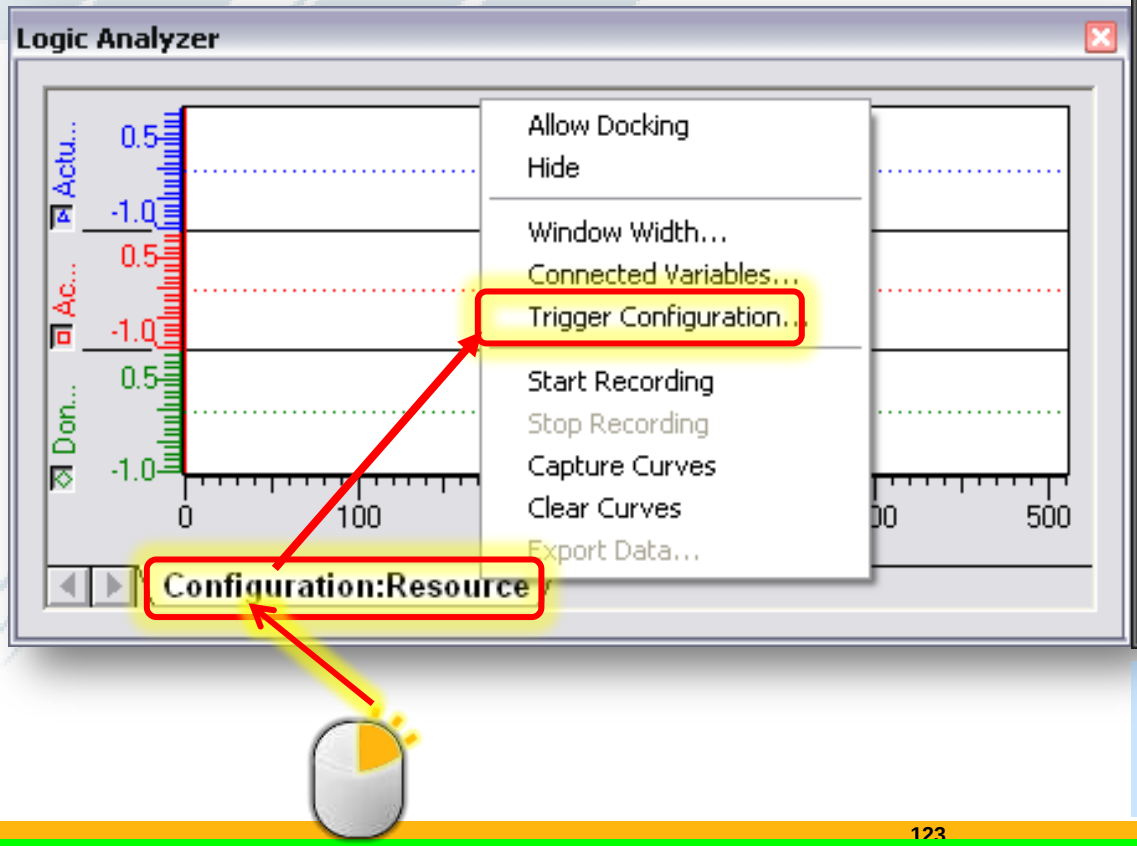
Add To Logic Analyzer

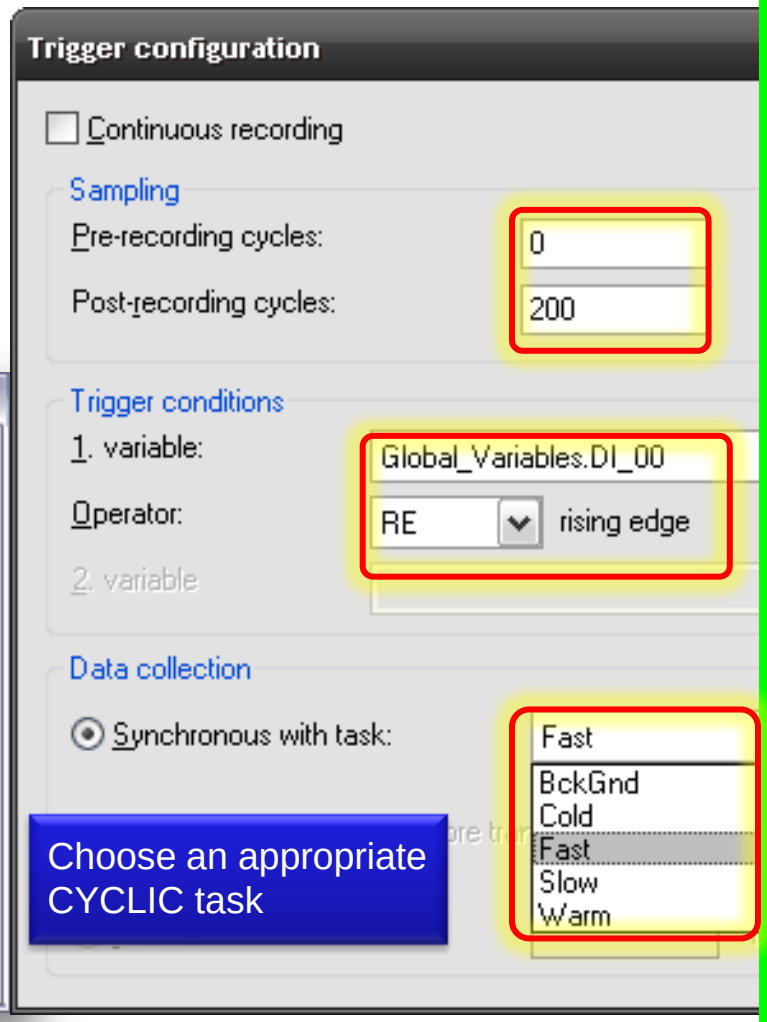
- G_NumberOfCars
- DI_00
- G_GateOpen
- DO_00



Trigger Configuration

- *Right-click Tab*
  - *Sampling*
  - *Trigger condition*
  - *Task*



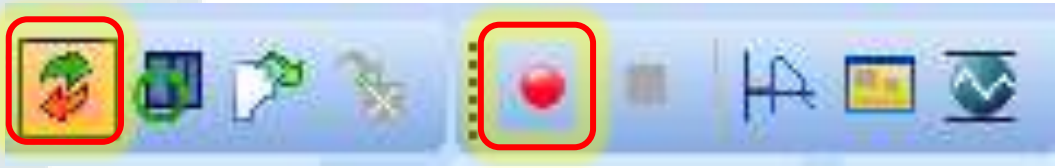


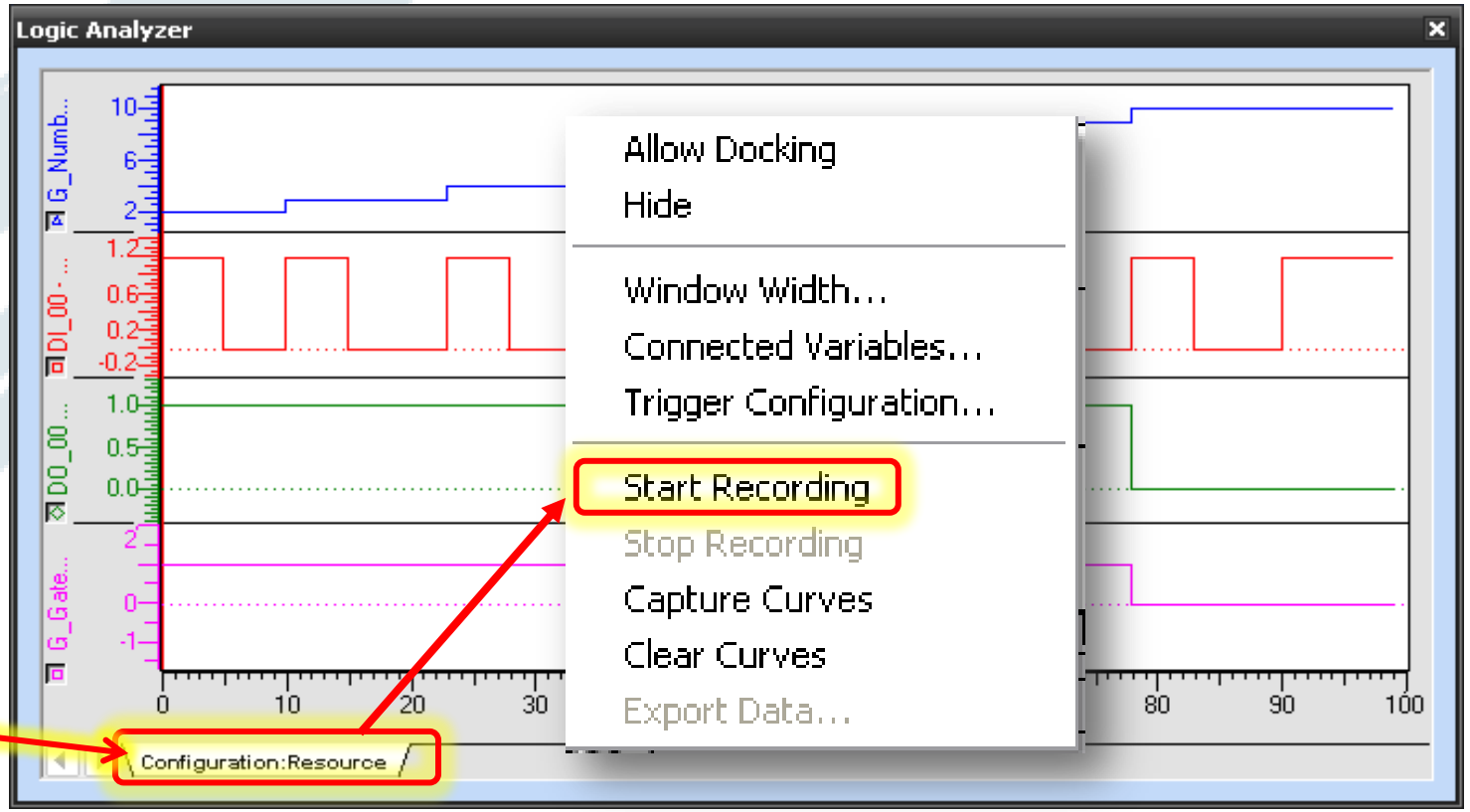
Data will be captured over how much time? (cycles… task time)

Logic Analyzer

Record Data

- *Trigger Configured*
- *Debug ON*
- *Start Recording*
- *Toggle DI_00*

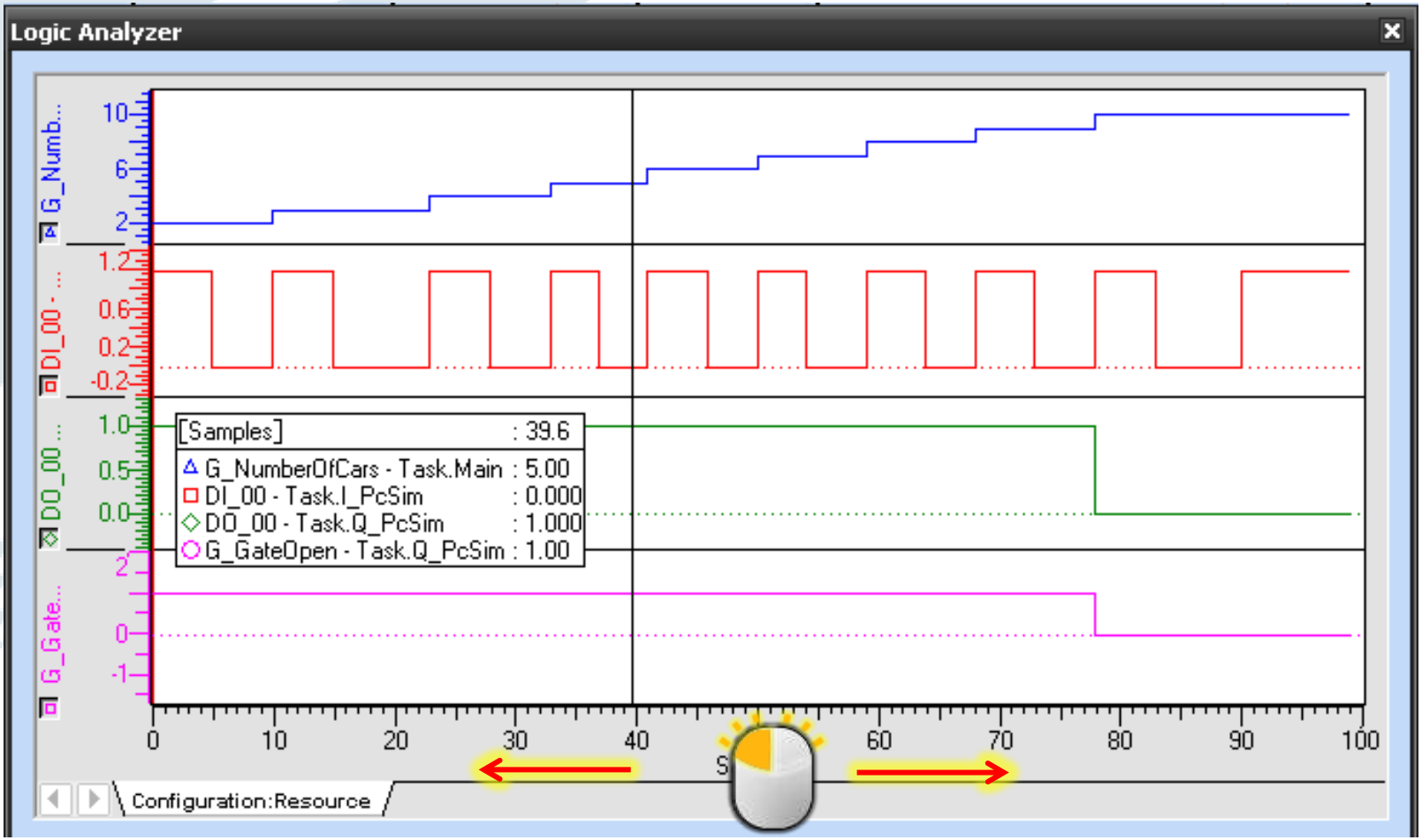




Logic Analyzer

Zoom

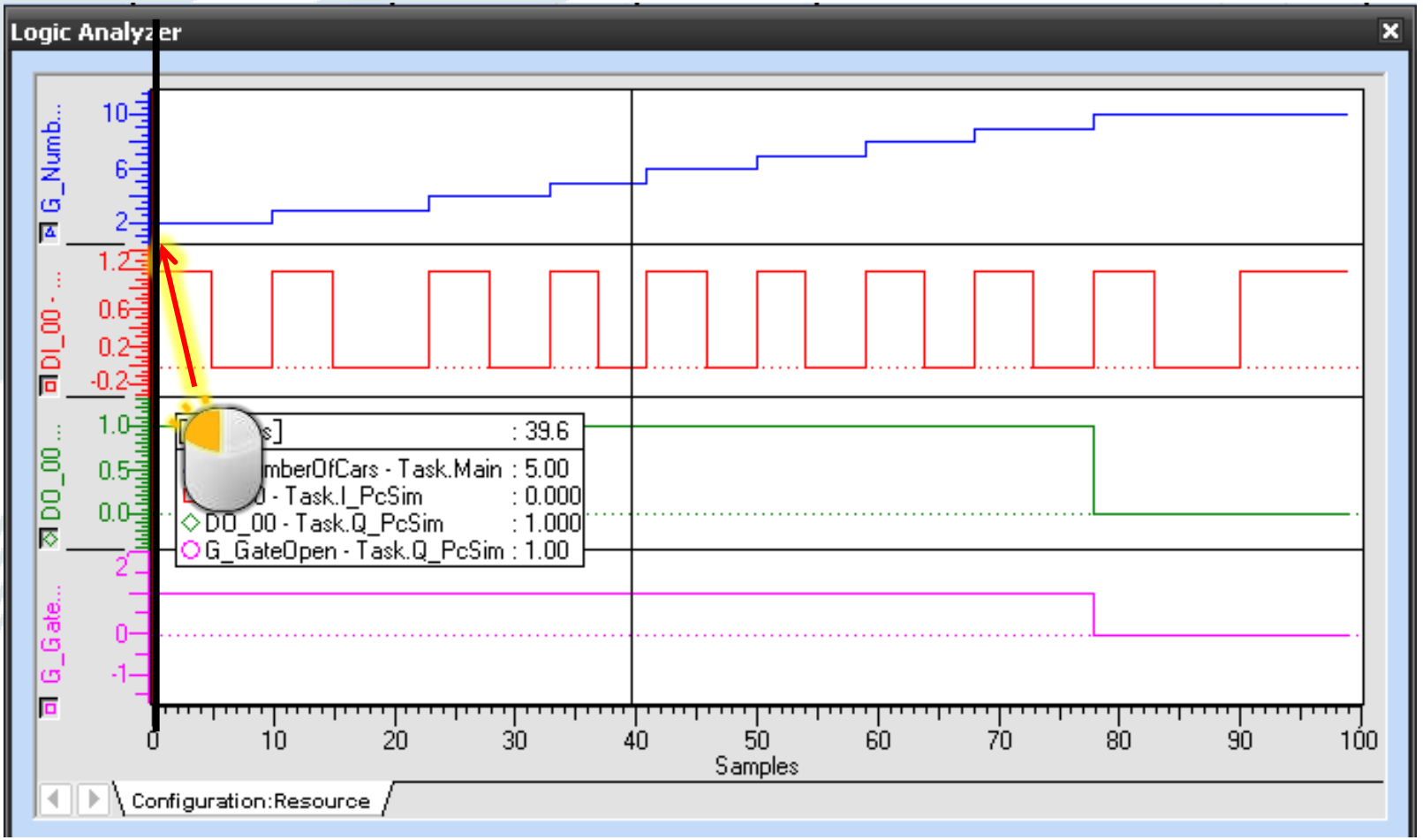
- *Mouse cursor over axis*
- *Right-Click and drag*
- *Left-Click and drag*



Cursors

- *One Vertical Cursor*
- *One Horizontal Cursor*

Verify the correct number of “cars” though DI_00 before gate closes.



Program Data Transfer

Compile

Download/Upload

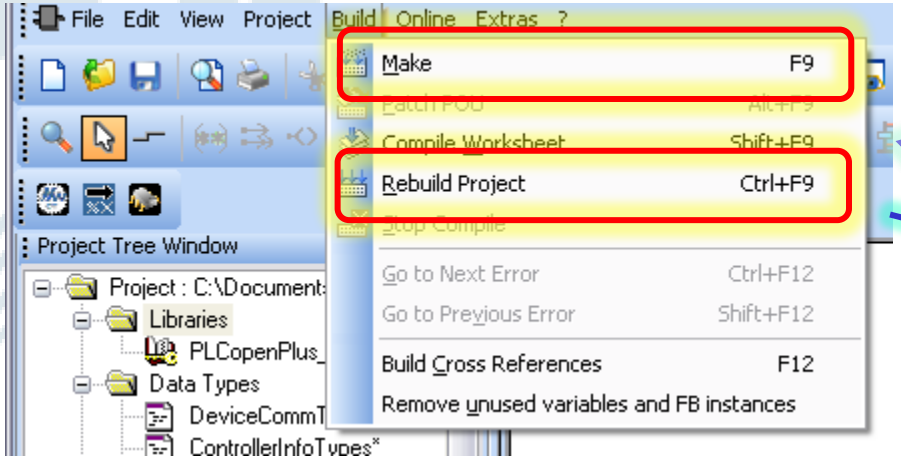
Cold/Warm/Hot Start

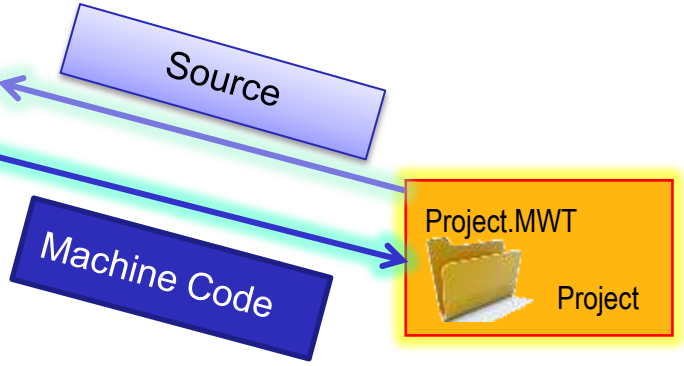
Power-On Sequence

Compile

- *"MAKE" (F9)*
  - *Saves project source*
  - *Compiles changes only*
  - *Faster*
  - *Normal Usage*

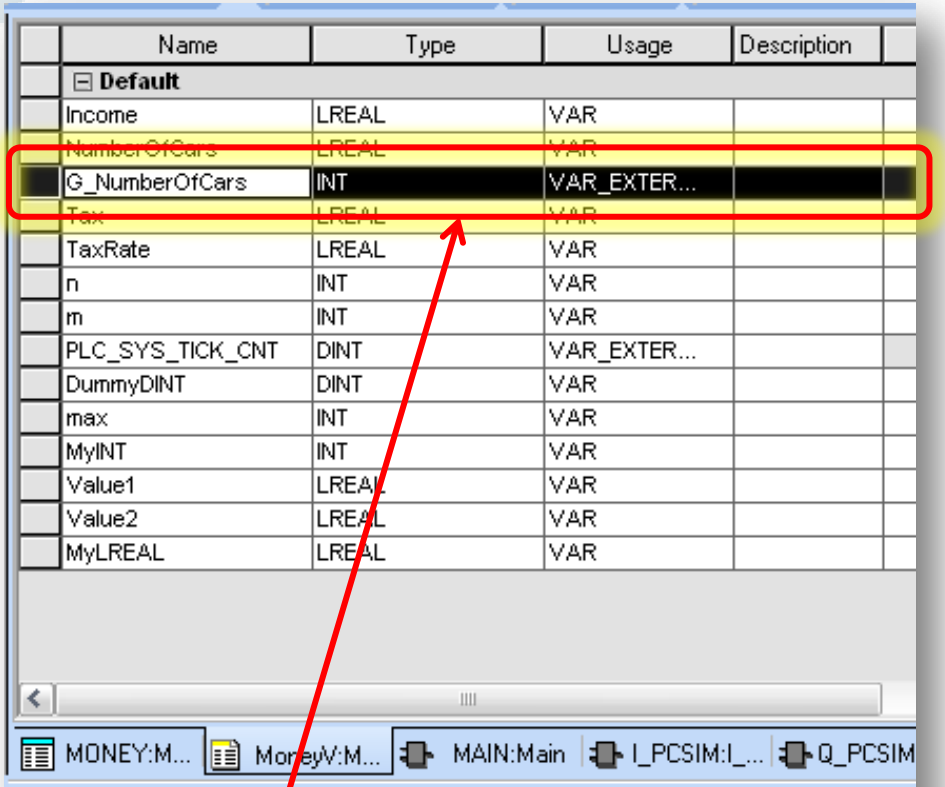
- *"Rebuild Project" (Shift+F9)*
  - *Saves project source*
  - *Compiles entire project*
  - *Slower*
  - *Periodic Usage*

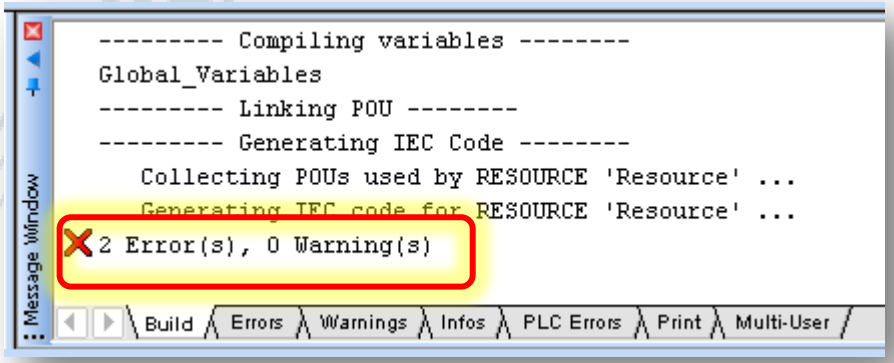


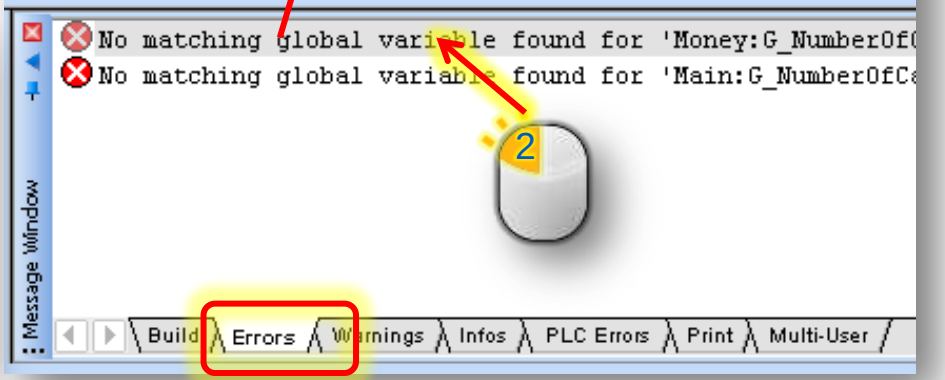


Compile Errors

- *Click Errors Tab*
- *Double-Click Error*
  - *Location of Error*
- *Fix Error*
- *Make*





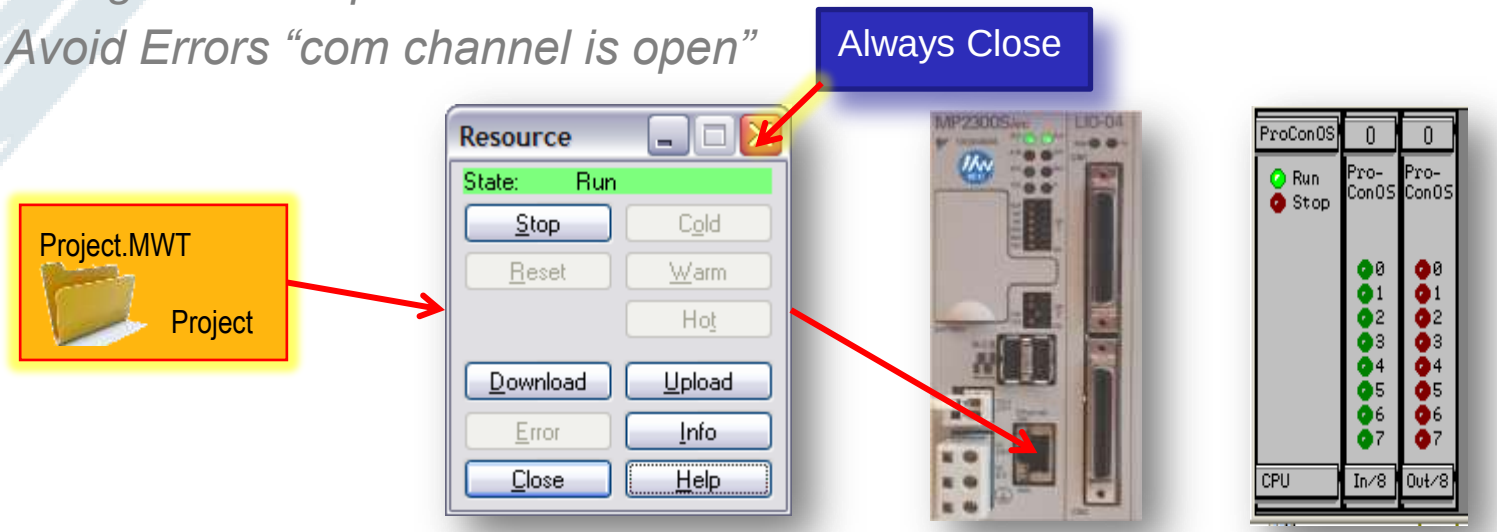


Download

Project Control Button

- Connection to the controller
  - » Normal “online” status is green
  - » If Error button appears red, click it to view/resolve error

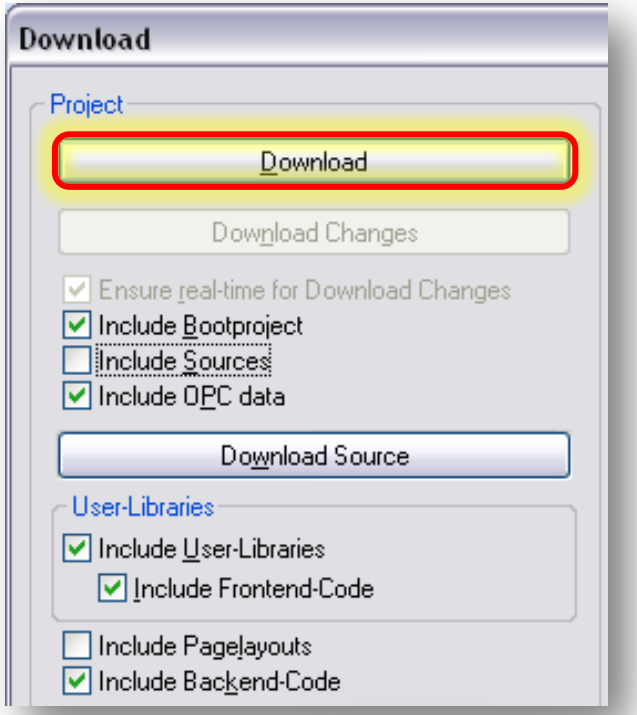
- Close the Resource Window
  - » Tasks cannot be deleted or changed while open
  - » Avoid Errors “com channel is open”

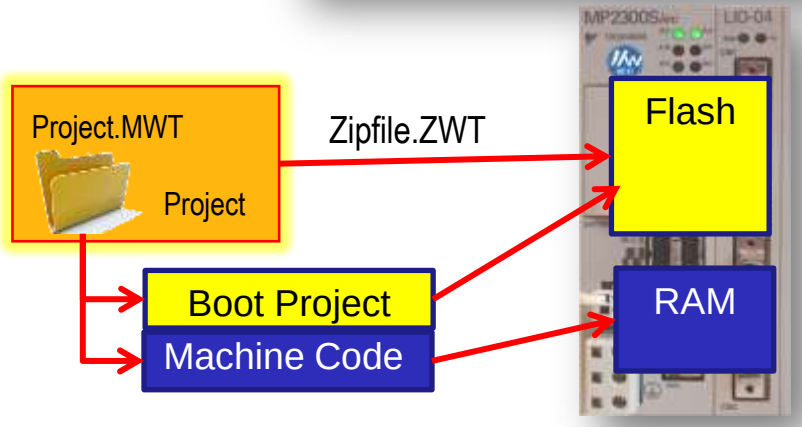


Download

- *Project*
  - *Machine code*
  - *Stored in RAM*
  - *Stop CPU*
- *Boot Project*
  - *Machine code*
  - *Stored in flash*
  - *Loads to RAM at power up*
  - *Executes at power up*
- *Sources*
  - *Zipped offline project folder*
  - *Stored to flash*
  - *Future upload*
  - *Download time slow*
  - *Intellectual property*

Uncheck items for faster download



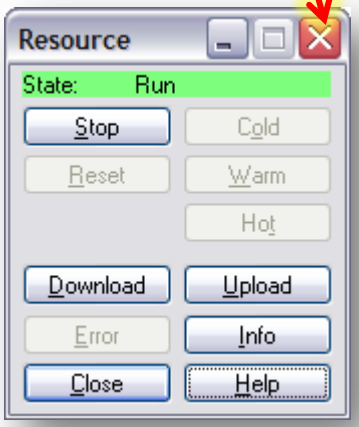


Download Changes

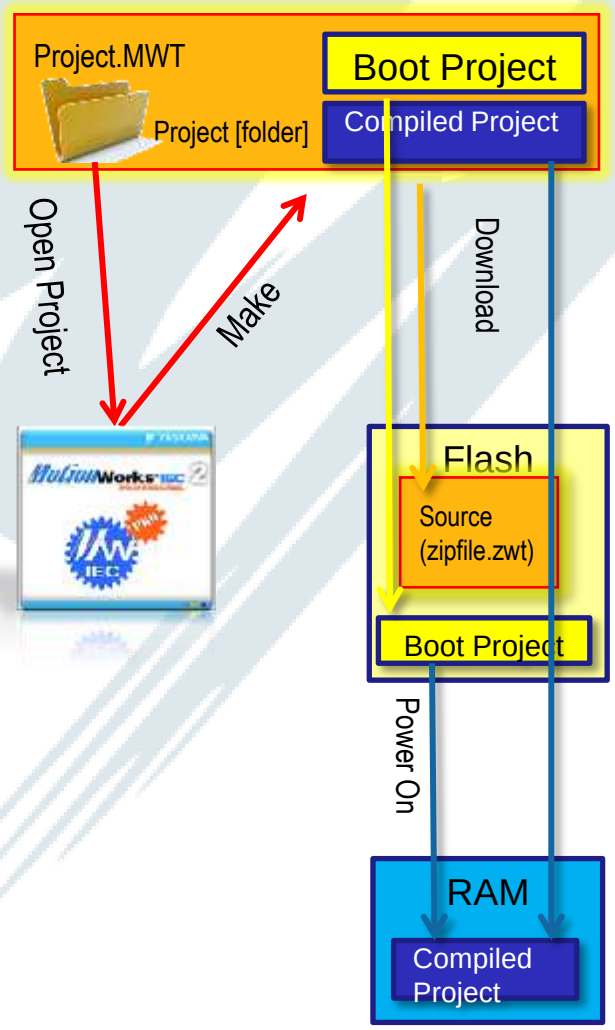
- *Download Changes Includes:*
  - *MAKE all edited POUs*
    - » *Automatic SAVE*
  - *Download Changes*
  - *CPU does not stop*
  - *Limitation (Not Supported)*
    - » *Hardware Change*
    - » *No project on controller*
    - » *Other*
- *Works by memory swap*
  - *Compiled project must be <1/2 PLC RAM capacity*

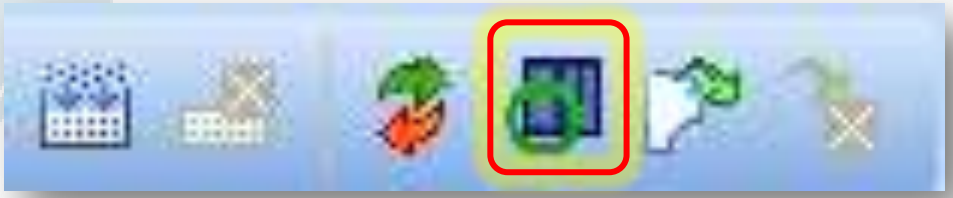
Download changes is FAST and EASY

Always Close

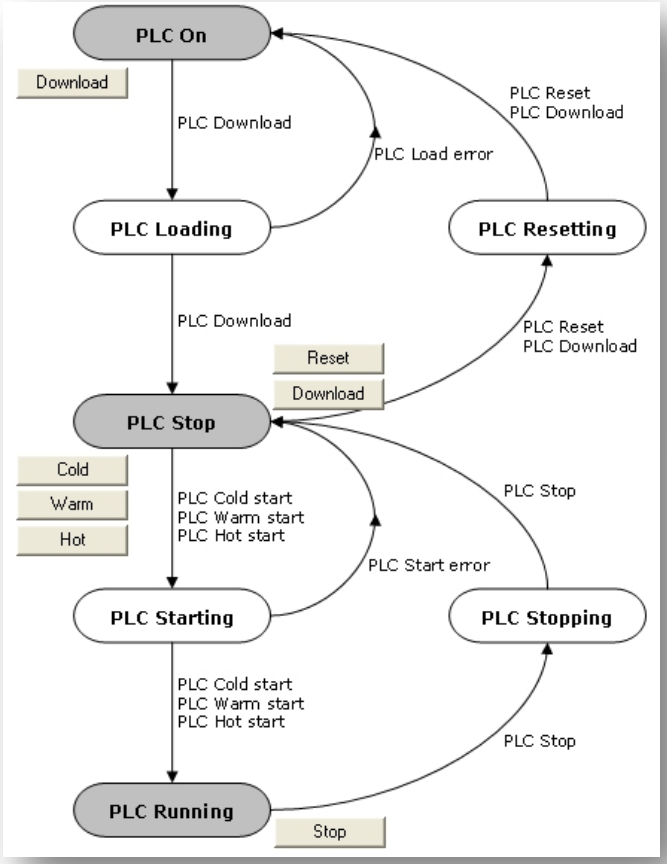


Program Data Transfer Diagram



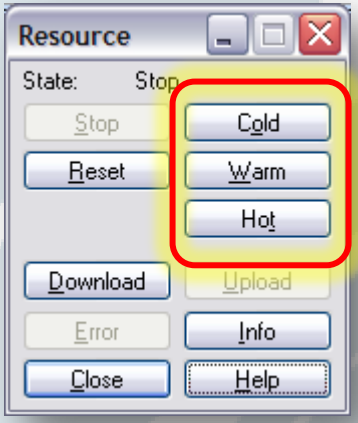


Project Control



Cold / Warm / Hot Start

Start CPU: Cold, Warm, or Hot



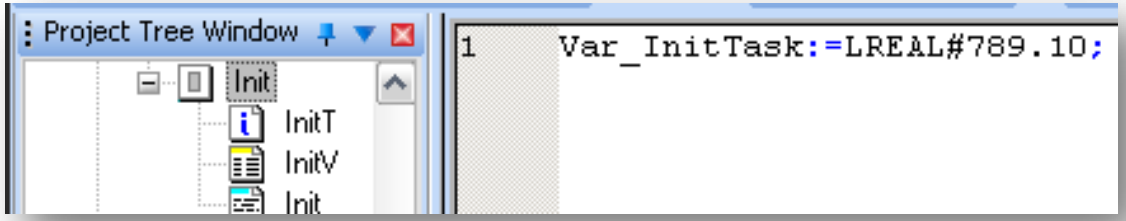
	Cold Start (Machine Commission)	Warm Start (Power On)	Hot Start
Retain Variables Initialized	Yes	No	No
Non-Retain Variables Initialized	Yes	Yes	No
ColdStart Task Runs	Yes	No	No
WarmStart Task Runs	No	Yes	No

Training Action - Program

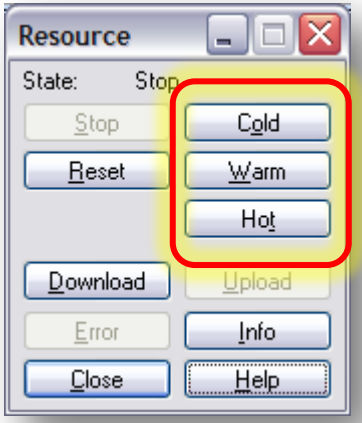
1. *Add variables to global list*

Name	Type	Usage	Init	Retain
System				
Var_Retain	INT	VAR_GLOBAL	123	☑
Var_NonRetain	INT	VAR_GLOBAL	456	☐
Var_InitTask	LREAL	VAR_GLOBAL		☐

2. *Add code to Init POU*

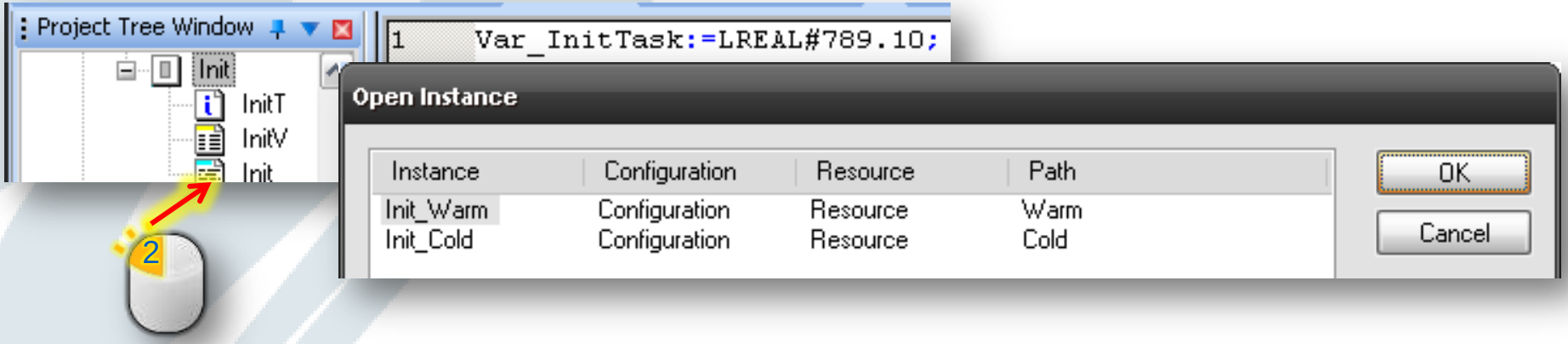


- *Download Changes*
- *Debug Mode*
- *Stop and start controller using Cold/Warm/Hot*



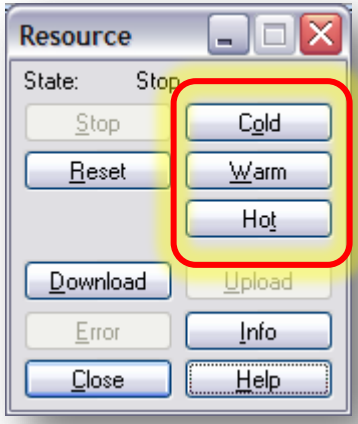
Training Action - Monitor

- *Program POU instance used in more than one task*



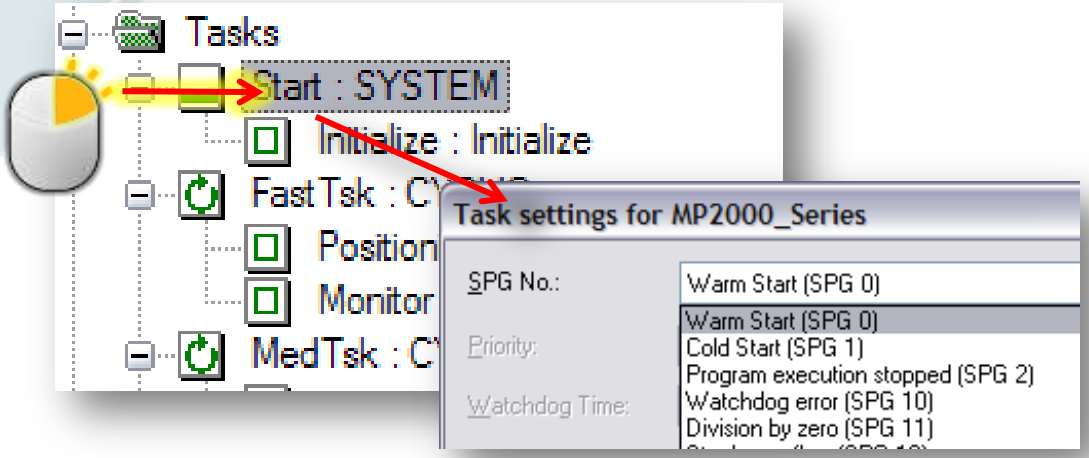
- *Change Online Values*
  - *Stop and start controller*

Name	Online value	Type	Usage	Init
System				
Var_Retain	123	INT	VAR_GLOBAL	123
Var_NonRetain	456	INT	VAR_GLOBAL	456
Var_InitTask	789.10	LREAL	VAR_GLOBAL	



Cold / Warm / Hot Start

- *Initialize Data at CPU Start*
  - *Variable "Init" Value*
  - *Warm Start Task*
  - *Cold Start Task*



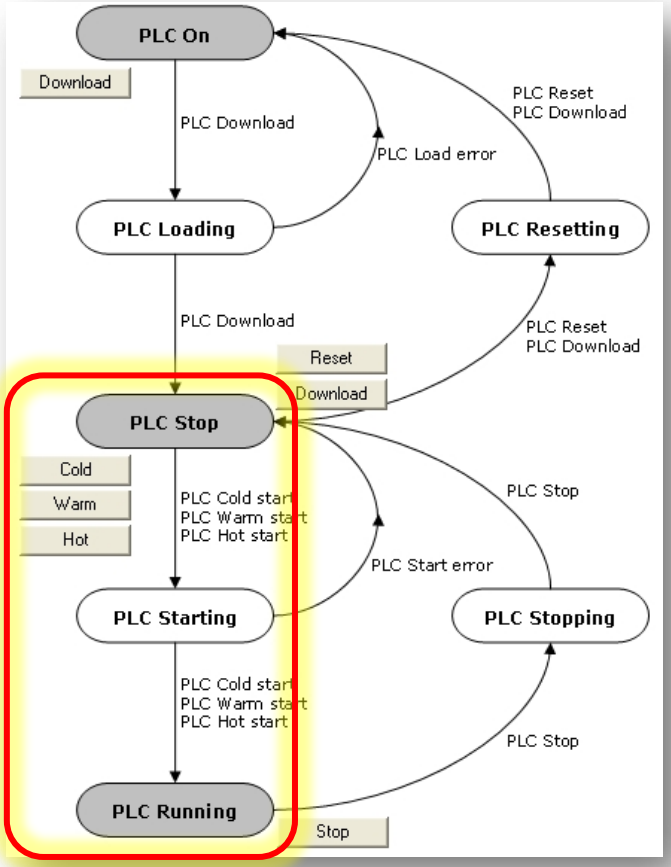
Name	Online value	Type	Usage	Description	Address	Init	Retain
Default							
HomeVel	55.000	LREAL	VAR			45.0	☐
HomeAccDec	360000.000	LREAL	VAR			360_000.0	☐
HomeOffsetLeft	-332.800	LREAL	VAR			20.0	☑
HomeOffsetRight	-126.000	LREAL	VAR			0.0	☑
HomePosition	0.000	LREAL	VAR			0.0	☐
G_AbsResetDoneRight	FALSE	BOOL	VAR_EXT...				☐

Why mark a variable "Retain"?

- Home Offsets, Part Counters and other variables have to keep their values when the power is off.
- The controller uses the lithium battery to keep these variable values "alive."

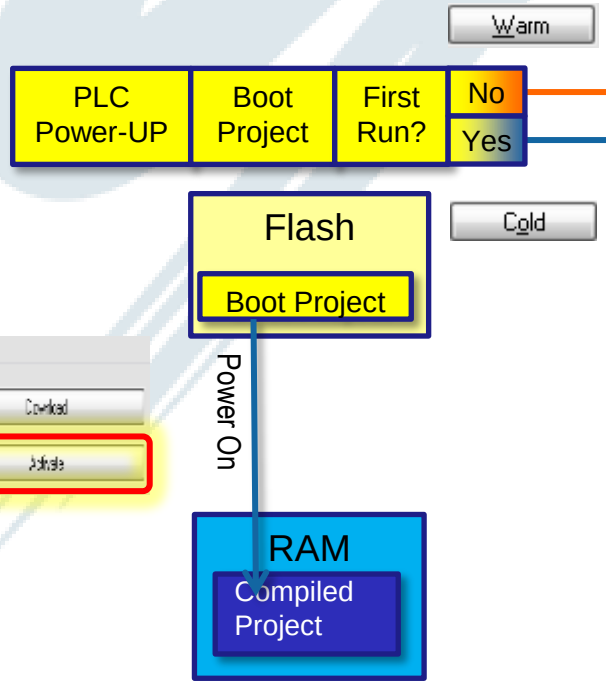
Power-On Sequence

- *Normal PLC Power-On Execution Sequence*
  - *Load Boot Project*
    - » *From FLASH to RAM*
  - *Warm Start*
    - » *Initialize non-retain variables*
    - » *Run Warm system task 1x*
    - » *Run Cyclic Tasks*
- *Simulator has no “power on”*
  - *Boot project is activated manually*
  - *Cold or warm is activated manually*



Program Data Transfer

Power On Timing



Warm Start

Warm : SYSTEM Task : CYCLIC

Name	Online value	Init	Retain
Var_Retain		123	☑
Var_NonRetain	←	456	☐

Cold Start

Cold : SYSTEM Task : CYCLIC

Name	Online value	Init	Retain
Var_Retain	←	123	☑
Var_NonRetain	←	456	☐

Variables

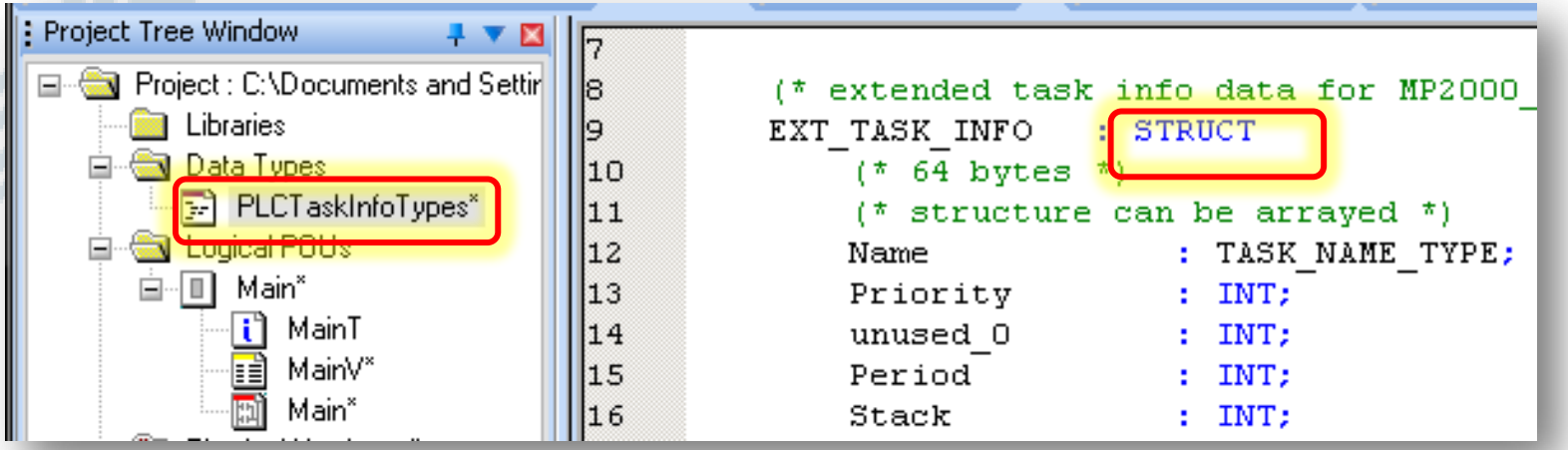
Name Rules
Data Type
Global/Local
Literal, Keyword
Variable Usage
Renaming Variables
Local and Global with same name
Variable List Organization
Global List Cleanup
Remove Unused Instances

YASKAWA

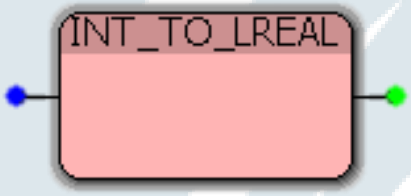
Variable Names

- *Alphanumeric characters and underscore "_"*
  - *Abc_123*
- *Capitalization is displayed but not important*
  - *Var1 = var1 =VAR1*
- *30 character maximum*
  - *Structures and "dot notation" for longer names*
    - » *Ex: FeedAxis.Home.CreepSpeed*
    - » *Declare structures in a Data Types worksheet*

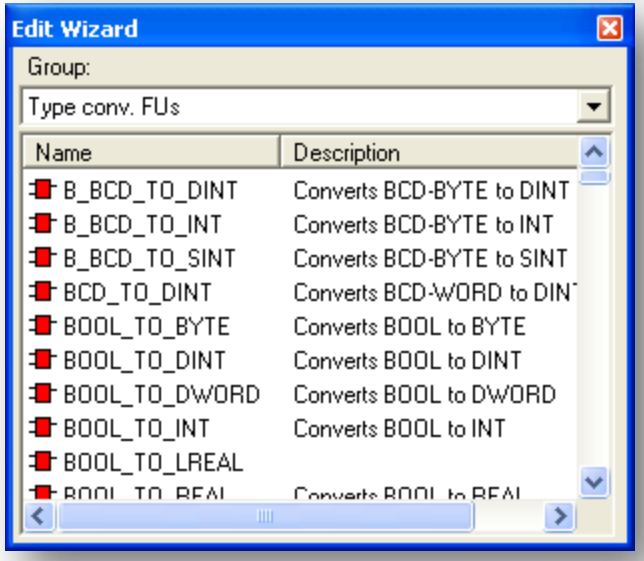
These rules and syntax are VERY strict.
There are no exceptions!



- *Convert the data type*
  - *Type Conversion Functions*
  - *No automatic conversion*




```
1        3.0000000 NumberOfCars :=INT_TO_LREAL( G_NumberOfCars );
2
3       89.9700000 Income:= LREAL#29.99 * NumberOfCars;
```

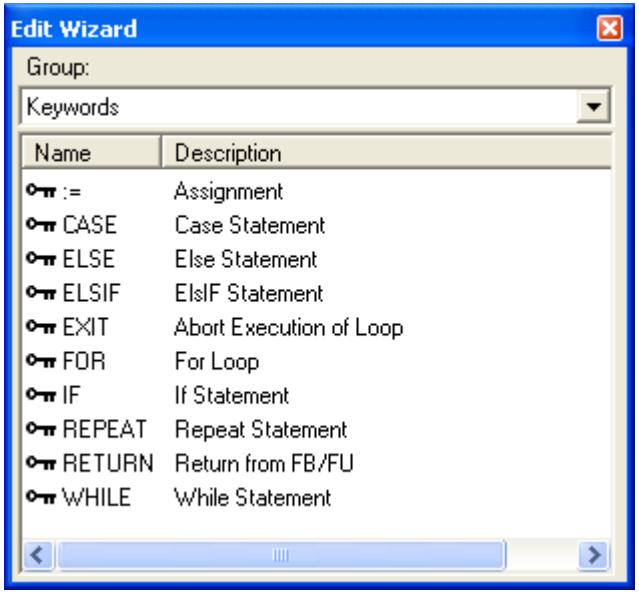


Literals & Keywords

- *Literals*
  - *REAL#1.0*
  - *INT#3*
  - *TIME#2s*
- *Keywords*
  - *List in "edit wizard"*
  - *Keywords only apply to ST POUs*

Notice the decimal point – never an "implied" decimal point

Notice ALL CAPS to define a literal value

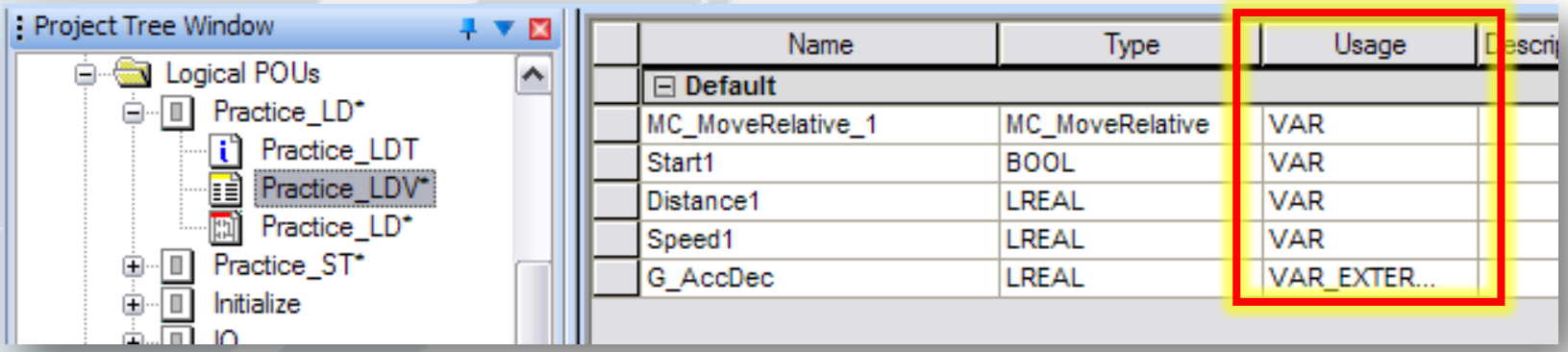


“ _ “ Underscore Character

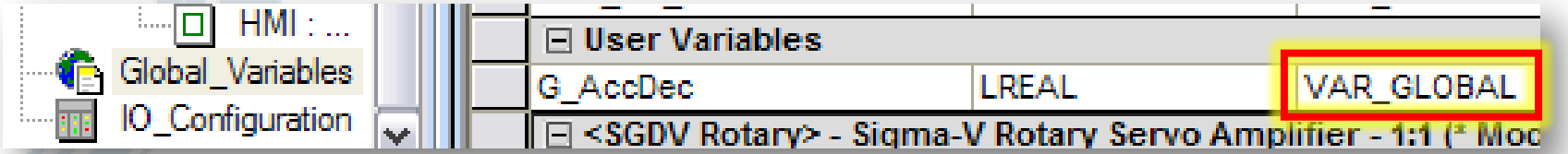
- *With Alphanumeric Characters in Variable Name*
  - *Servo_01*
  - *_01Servo*
- *Place Holder Within Data (Cosmetic Only)*
  - *Initial Values*
    - » *100_000_000.0*
  - *Literals*
    - » *LREAL#100_000_000.0*

View Variables

- *View Local Variables*



- *View Global Variables*



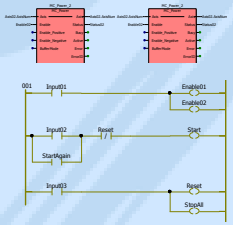
Variable Usage

POU 1

Variables: POU 1

Name	Usage
V001	VAR
V002	VAR
G_AccDec	VAR_EXTERNAL
G_V004	VAR_EXTERNAL

POU 2



Variables: POU 2

Name	Usage
V001	VAR
V002	VAR
G_AccDec	VAR_EXTERNAL
G_V005	VAR_EXTERNAL

POU xxx

Variables: POU xxx

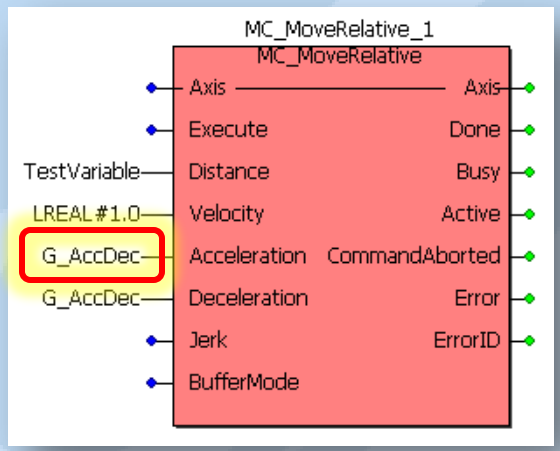
GLOBAL_VARIABLES

Name	Usage
G_AccDec	VAR_GLOBAL
G_V004	VAR_GLOBAL
G_V005	VAR_GLOBAL

Global Variables

- *Create global and local variables*

Practice_LD



Variables: Practice_LD

Name	Type	Usage
Default		
MC_MoveRelative_1	MC_MoveRelative	VAR
G_AccDec	LREAL	VAR_EXTER...

Best Practice:
Name global variables with prefix G_

GLOBAL_VARIABLES

Name	Type	Usage
MO1_DI_15	BOOL	VAR_GLOBAL
MO1_DO_00	BOOL	VAR_GLOBAL
MO1_DO_01	BOOL	VAR_GLOBAL
MO1_DO_02	BOOL	VAR_GLOBAL
MO1_DO_03	BOOL	VAR_GLOBAL
MO1_DO_04	BOOL	VAR_GLOBAL
MO1_DO_05	BOOL	VAR_GLOBAL
MO1_DO_06	BOOL	VAR_GLOBAL
MO1_DO_07	BOOL	VAR_GLOBAL
MO1_DO_08	BOOL	VAR_GLOBAL
MO1_DO_09	BOOL	VAR_GLOBAL
MO1_DO_10	BOOL	VAR_GLOBAL
MO1_DO_11	BOOL	VAR_GLOBAL
MO1_DO_12	BOOL	VAR_GLOBAL
MO1_DO_13	BOOL	VAR_GLOBAL
MO1_DO_14	BOOL	VAR_GLOBAL
MO1_DO_15	BOOL	VAR_GLOBAL
User Variables		
G_AccDec	LREAL	VAR_GLOBAL

Global_Varia...

Practice_ST

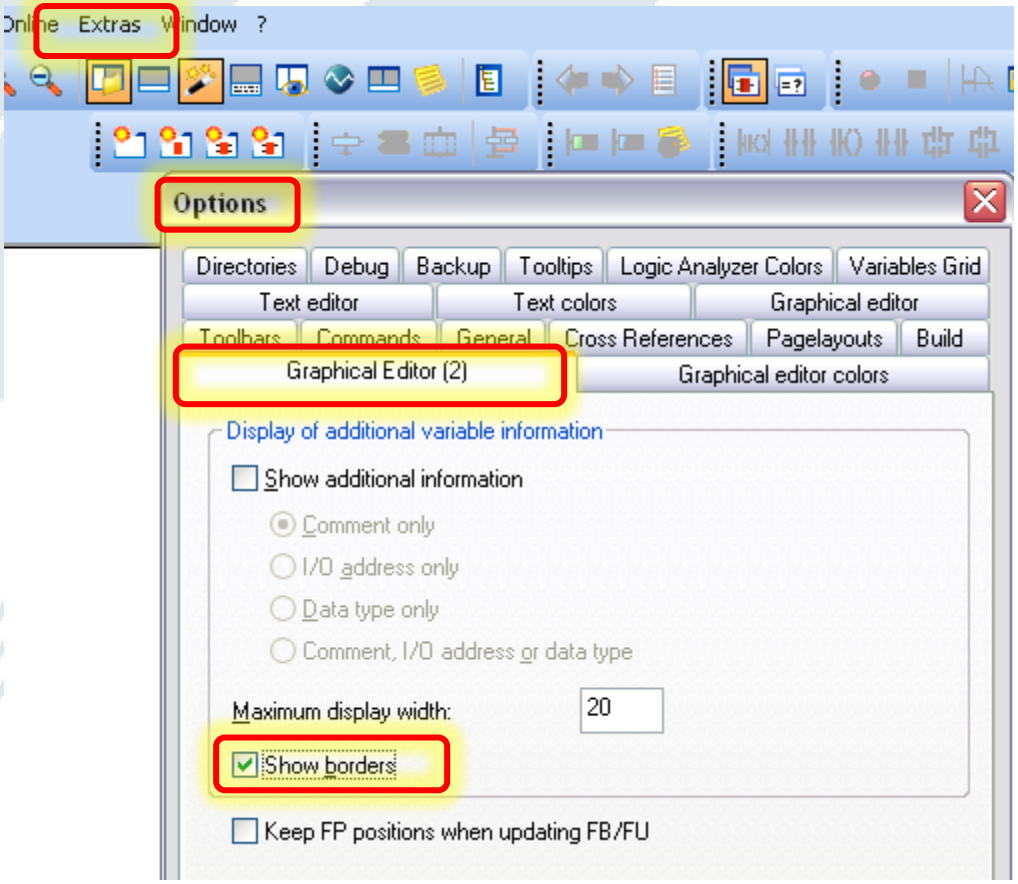
```
MC_MoveRelative_1(Axis:=
(* ANY *),Execute:=(* BOOL *),
Distance:= TestDistance (* LREAL *),
Velocity:= LREAL#1.0 (* LREAL *),
Acceleration:= G_AccDec (* LREAL *),
Deceleration:= G_AccDec (* LREAL *),
Jerk:=(* LREAL *),
BufferMode:=(* INT *));
(* ANY *):=MC_MoveRelative_1.Axis;
```

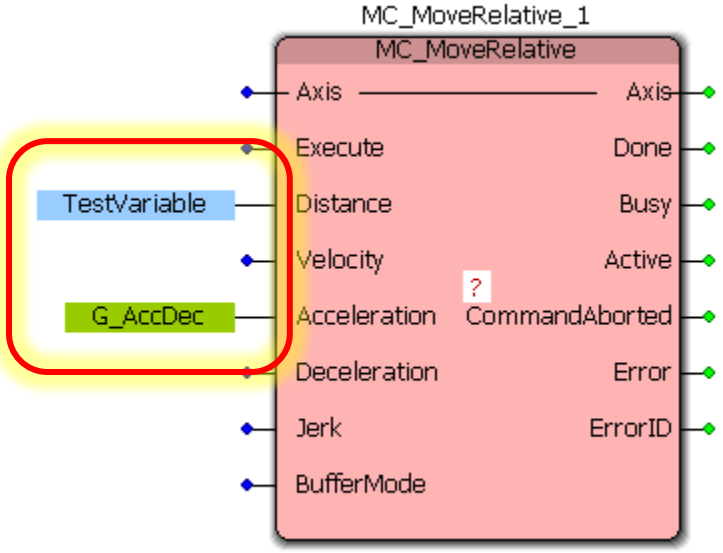
Variables: Practice_ST

Name	Type	Usage
Default		
MC_MoveRelative_1	MC_MoveRelative	VAR
TestDistance	LREAL	VAR
G_AccDec	LREAL	VAR_EXTERNAL

Global & Local Color Format

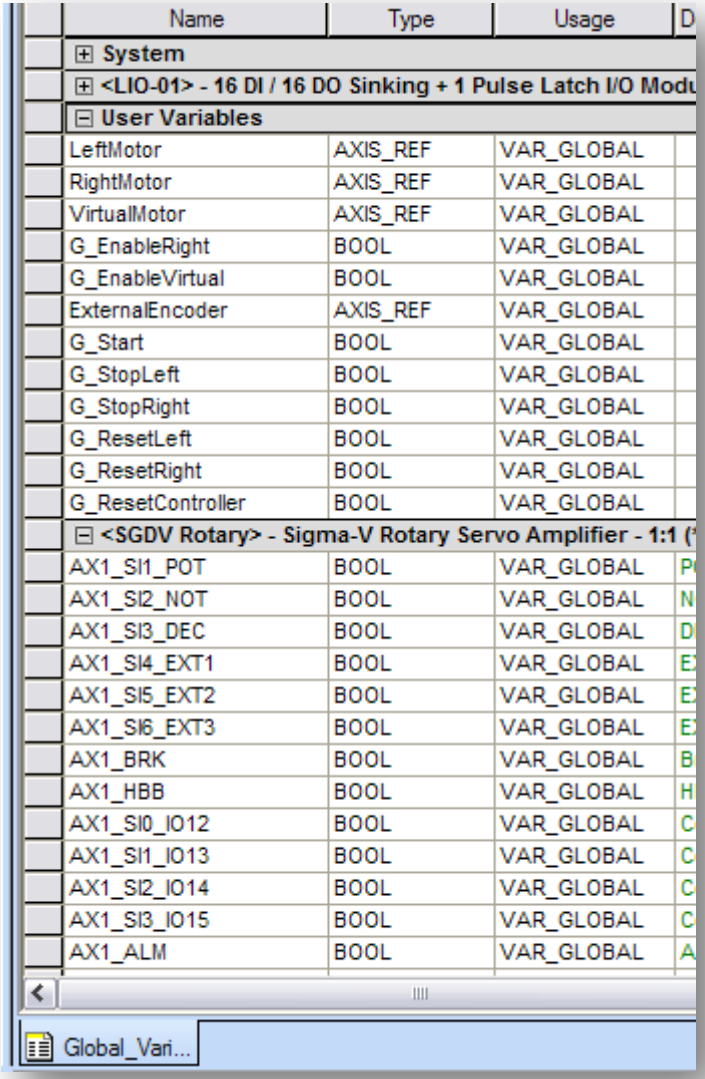
- *Use “Show Borders” for color confirmation of global and local*
  - *Extras, Options, Graphical Editor (2)*





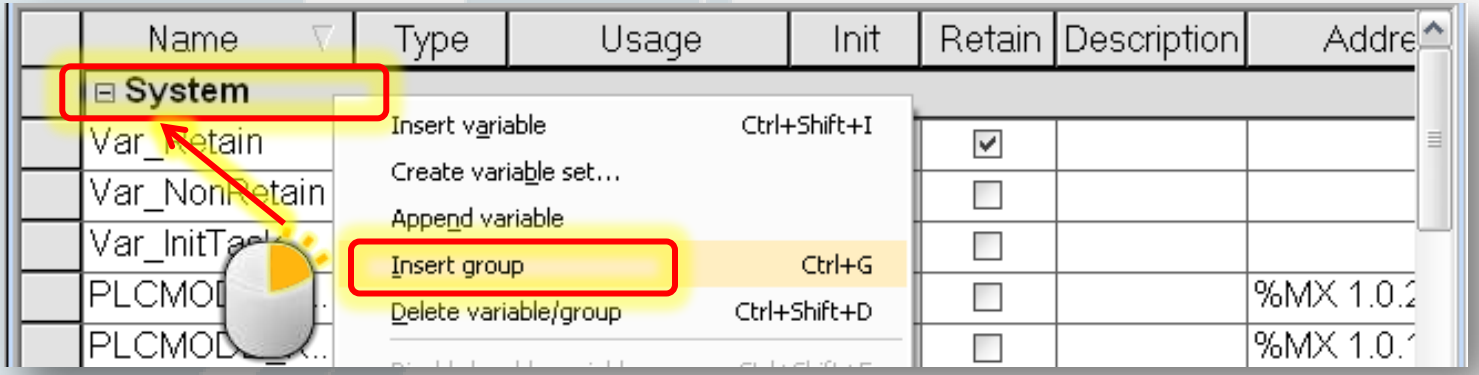
Variable Conventions

- *GLOBAL or LOCAL?*
  - *Use LOCAL unless global is required*
  - *Suggested Convention*
    - » *Type G_ in front of all global variables*
      - – *G_EnableRight*
      - – *G_EnableLeft*
    - » *Visual Confirmation when looking at global list*

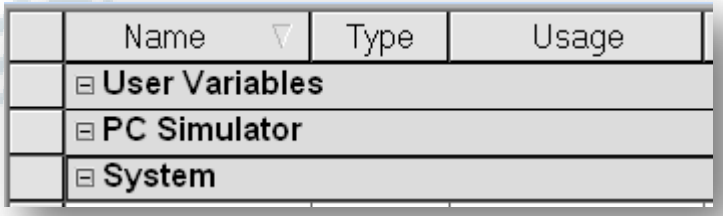


Variable Groups

- *Insert Variable Groups*



- *Rename*
  - *User Variables*
  - *PC Simulator*



Variable Groups facilitate organization of the global variables

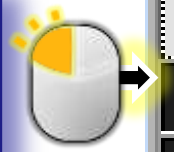
Variable List Organization

- *Global variables in the wrong group?*
  - *Example: Check LIO-01 Group*
  - *Drag variables to appropriate group*

Click and drag row header to highlight multiple rows

Click and drag row header **again** to move the group of variables

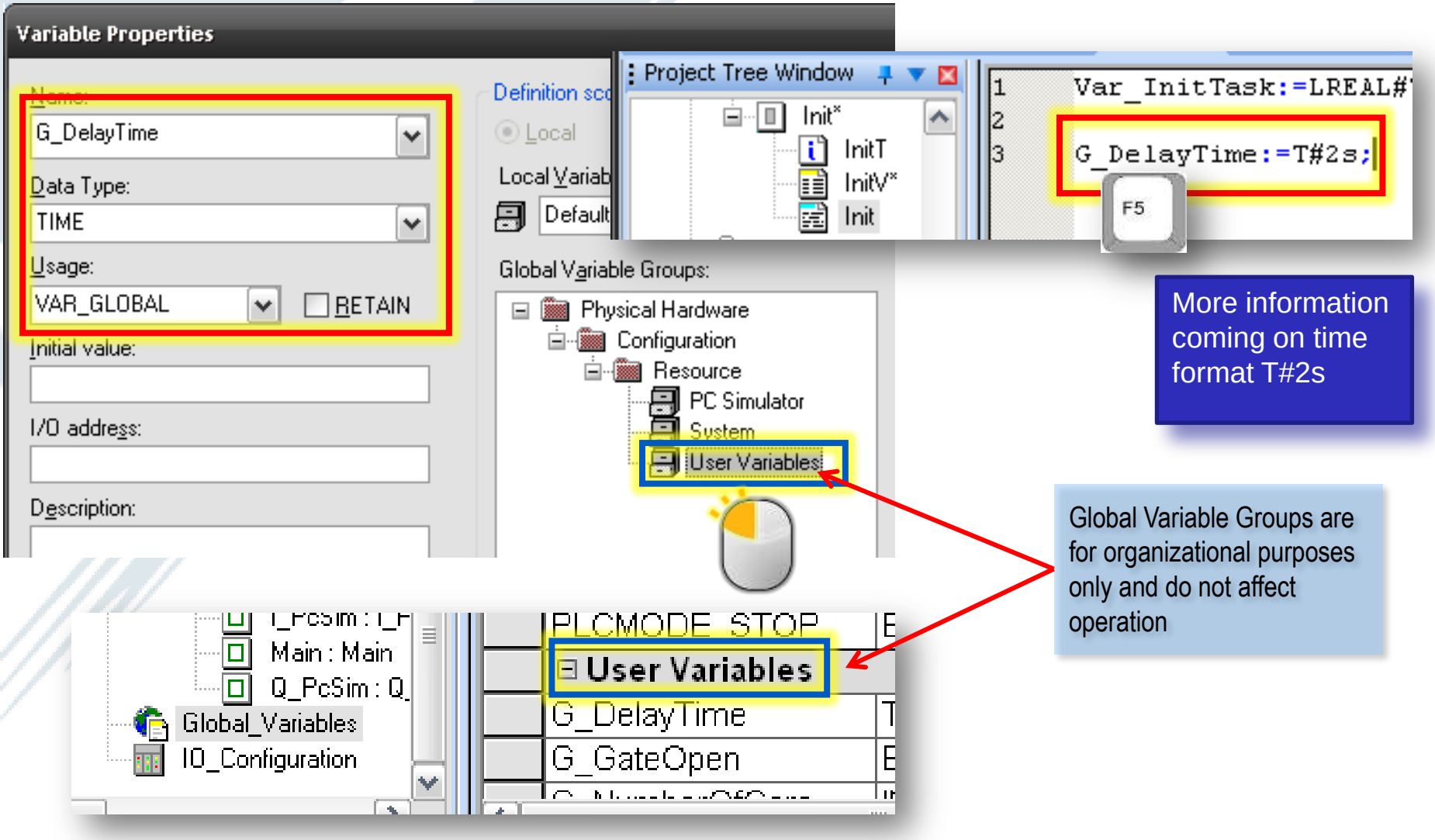
Red Line appears while dragging



Name	Type	Usage
⊟ **User Variables**		
⊟ **PC Simulator**		
⊟ **System**		
Var_Retain	INT	VAR_GLOBAL
Var_NonRetain	INT	VAR_GLOBAL
V…InitTas…	LREAL	VAR_GLOBAL
P…MODE_S…	BOOL	VAR_GLOBAL
PLCMODE_R…	BOOL	VAR_GLOBAL

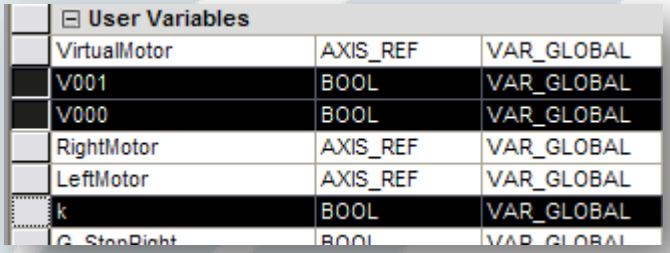
Variable Groups

- *Create the variable "G_DelayTime" in the Init POU*



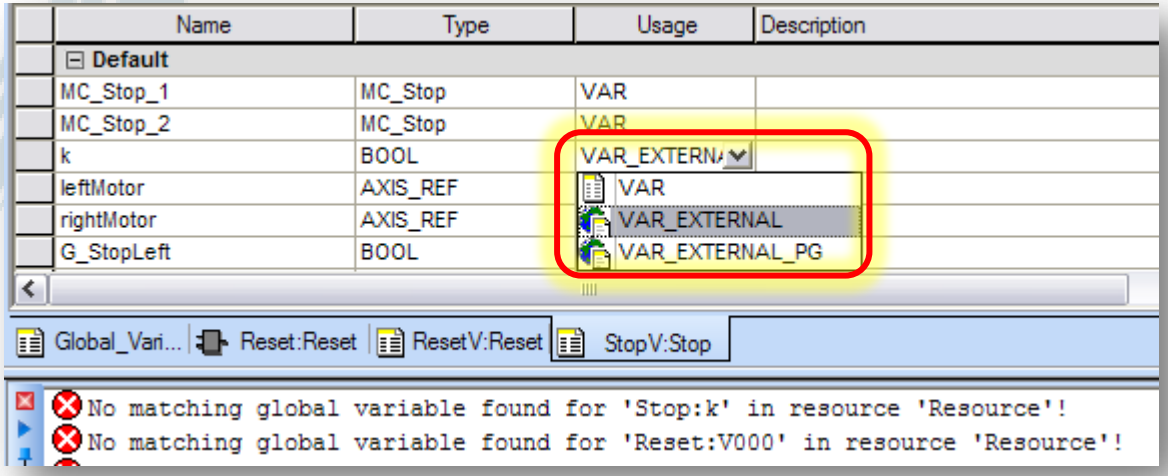
Global List Clean-up

- *Global List Has Mistakes*
  - *Delete them*



User Variables		
VirtualMotor	AXIS_REF	VAR_GLOBAL
V001	BOOL	VAR_GLOBAL
V000	BOOL	VAR_GLOBAL
RightMotor	AXIS_REF	VAR_GLOBAL
LeftMotor	AXIS_REF	VAR_GLOBAL
k	BOOL	VAR_GLOBAL
G_StopRight	BOOL	VAR_GLOBAL

  - *Change local list usage to "var"*
    - » *Find using MAKE*

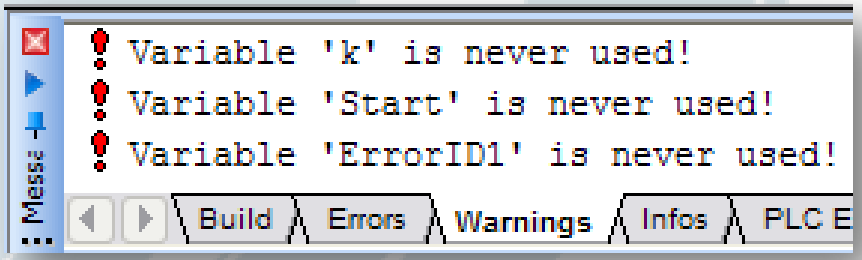


TIP: If you have a lot of variables to update…

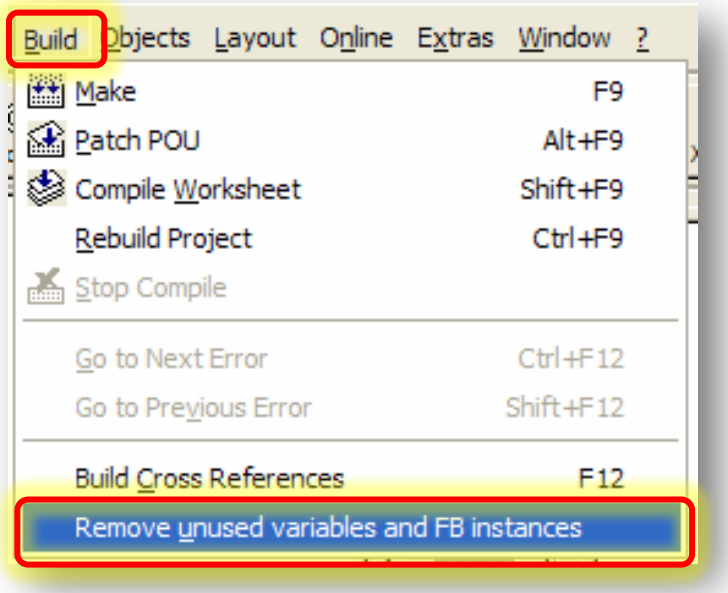
1. Click on the Usage field
2. Type the letter 'v'
3. Click the next one

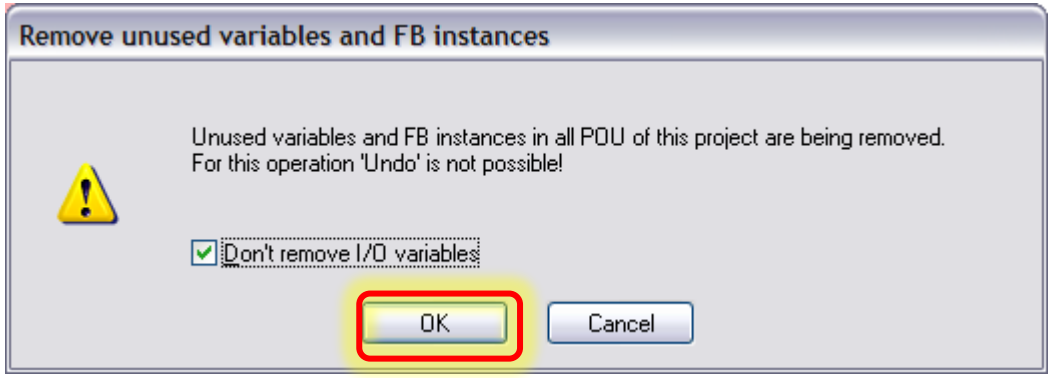
Remove Unused Variables

- *Warning – Variable is never used!*



  - *Build menu*
    - » *"Remove unused variables and FB instances"*
      - – *"Unused" means not used in a POU*
    - » *MAKE completes with no warnings*
    - » *Global variables NOT removed*



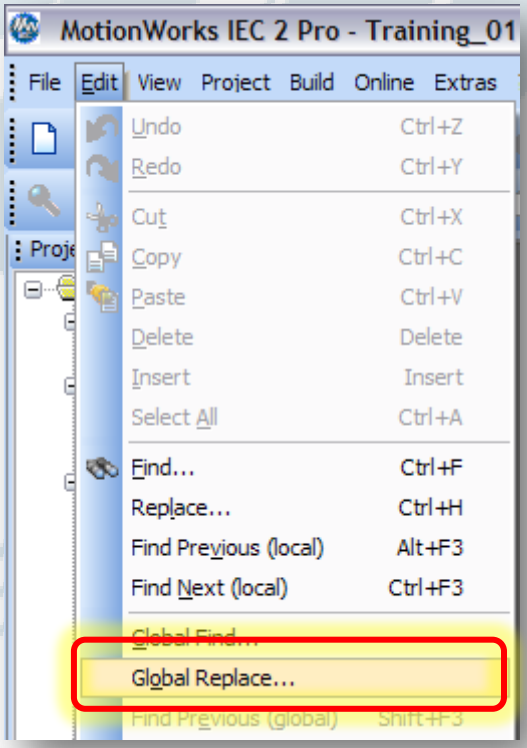


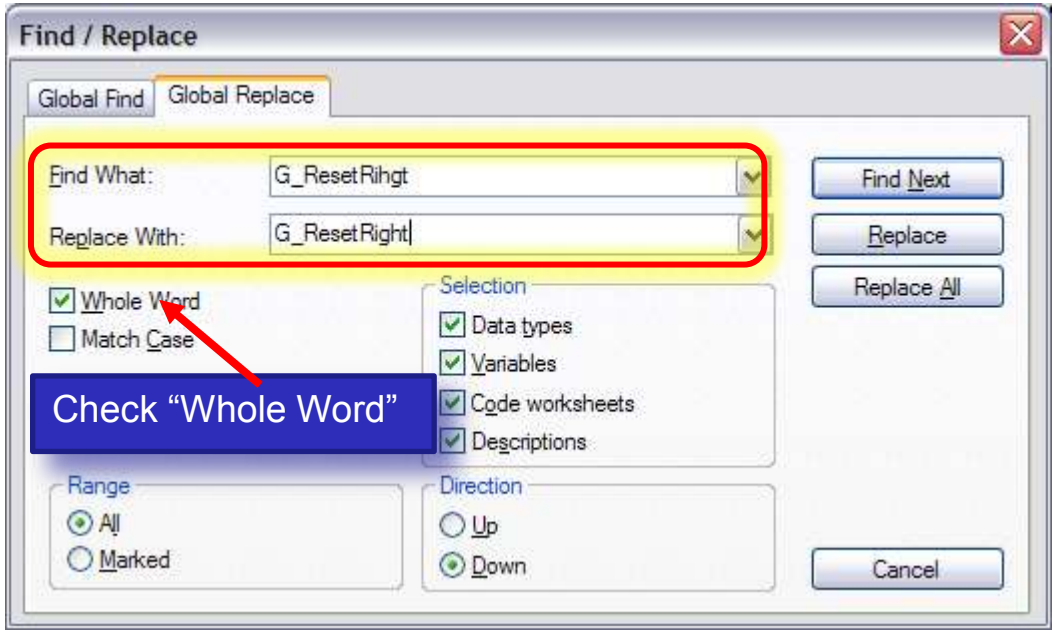
Renaming Variables

Rename Variables using Global Replace

- *Same name required*
  - » *Local Variable List*
  - » *Usage in Code*

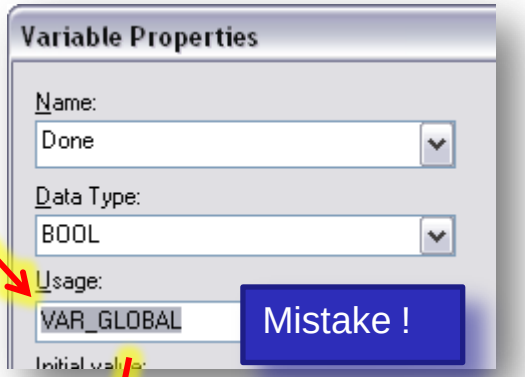
EXAMPLE:
Rename due to spelling error or naming convention

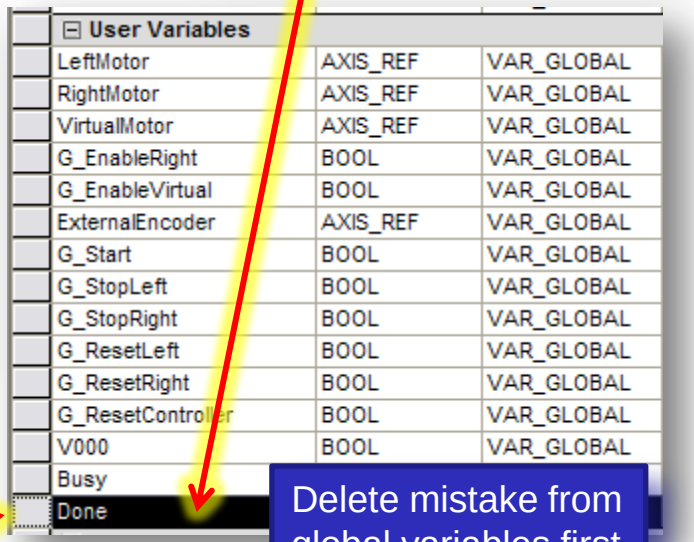




Local and Global Same Name

- *Common Mistake*
  - *Create global variable (without G_)*
    - » *Mistake – wanted local variable*
  - *Create local variable with same name*
    - » *CAUTION "New Local" or "Use Global"*





- *Solution*
  - » *Cancel the CAUTION*
  - » *DELETE global variable*

Driving Value

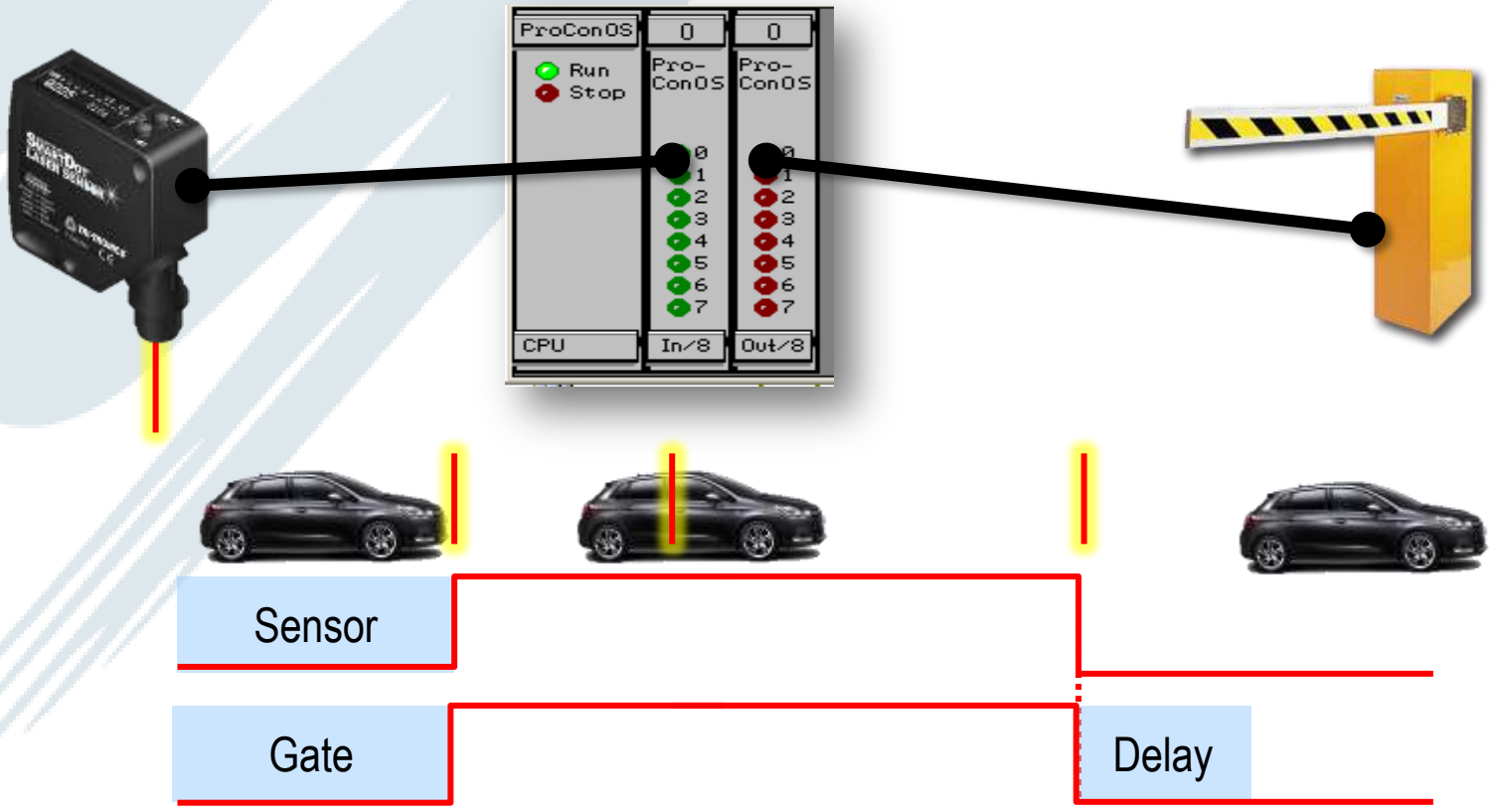
Timers and Sequencing

Overview
IEC 61131-3 Timers
Application
Sequencing

YASKAWA

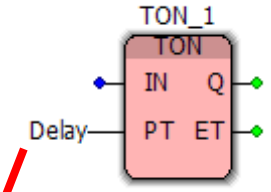
Timer Application Overview

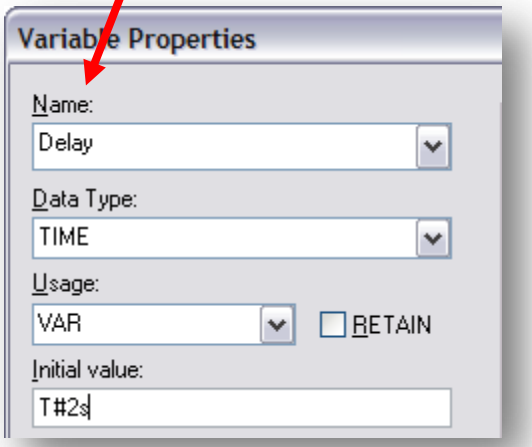
- *Application*
  - *Timer to close gate after sensor releases*

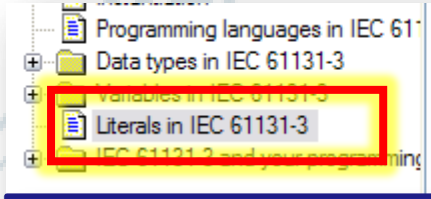


Using Timer

- *Right-click Help for parameter definition*
- *TIME data type has a special format*
  - *Initial values must be expressed as a literal*
  - *CORRECT*
    - » *T#5s*
    - » *T#5.0s*
    - » Time#5s
  - NOT CORRECT
    - » 5
    - » 5.0







Help File
MPReadMe001.chm

Duration literals

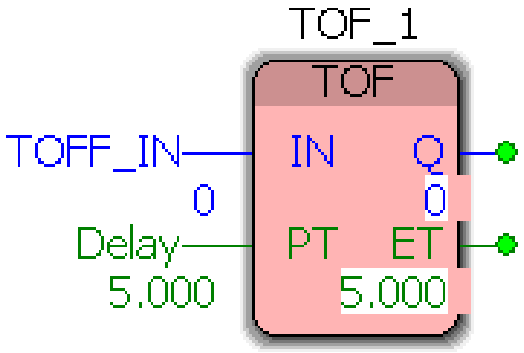
Duration data can be represented in hours, minutes, seconds, milliseconds and in combination of these formats.

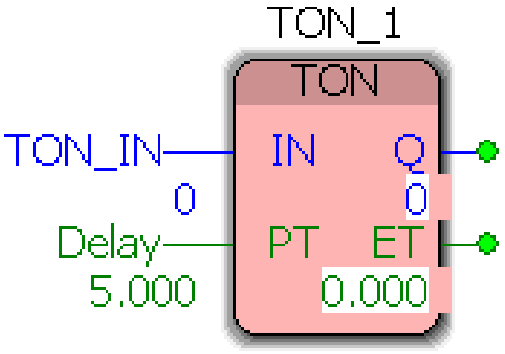
Type	Examples
Short prefix	T#14ms t#14ms t#12m18s3.5ms T#25h_15m t#25h_15m
Long prefix	TIME#14 ms time#14ms TIME#25h_15m time#25h_15m

IEC 61131-3 Timers

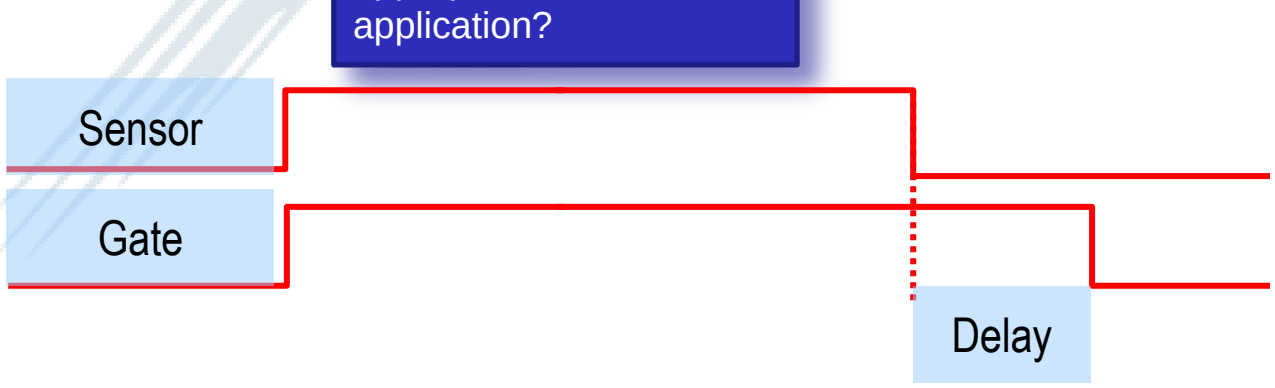
- *Create new LD POU "Timers"*
- *Run in Fast task*
- *Add each timer*

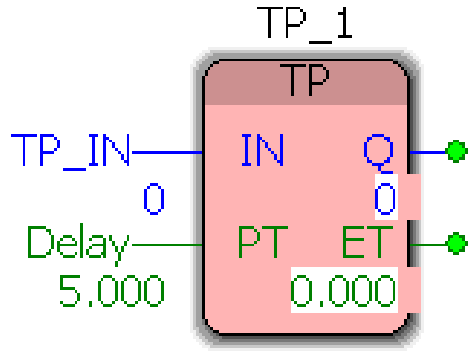
What happens if you change the value of IN during operation?





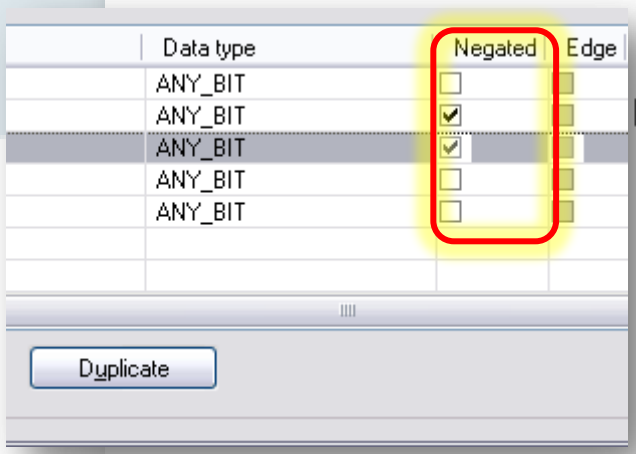
Which timer is appropriate for the application?

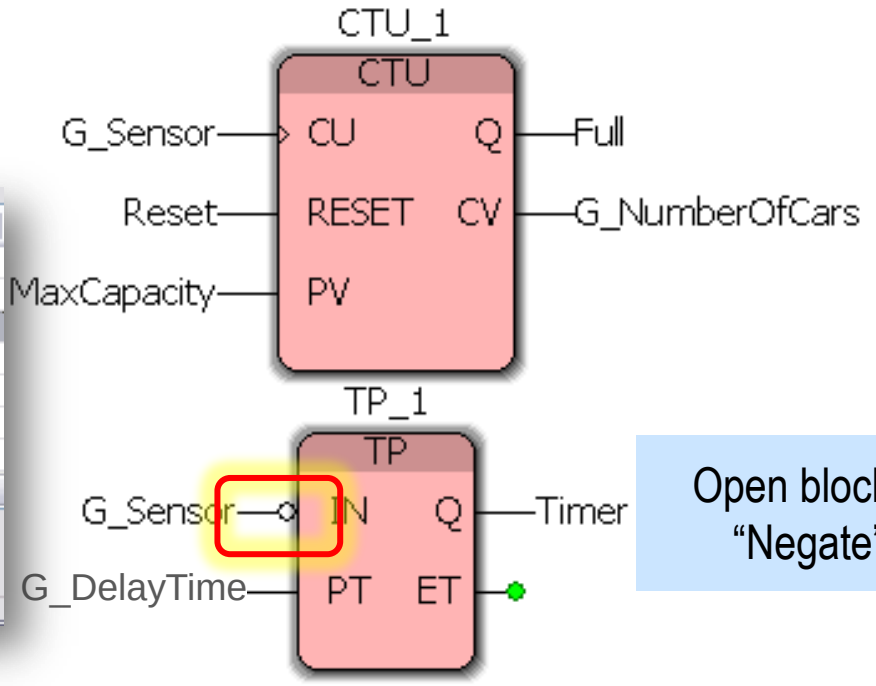




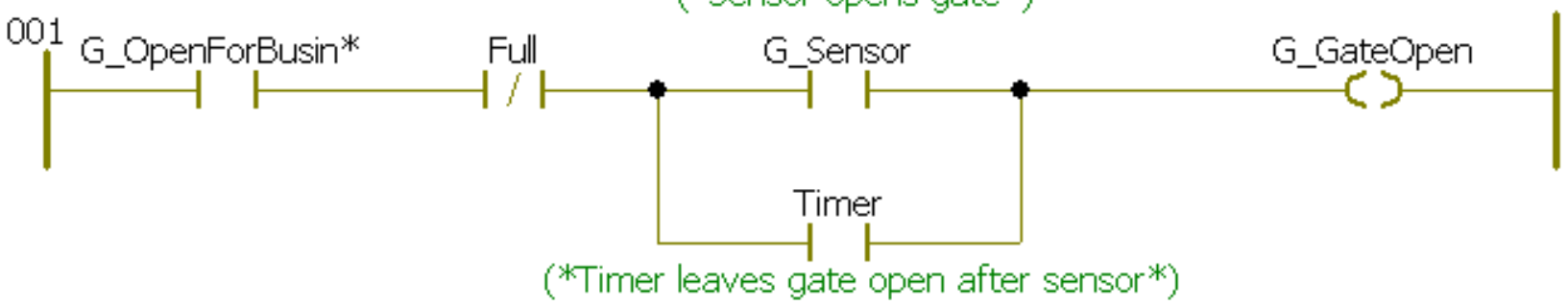
Timer Application

- *Implement timer modification to MAIN*



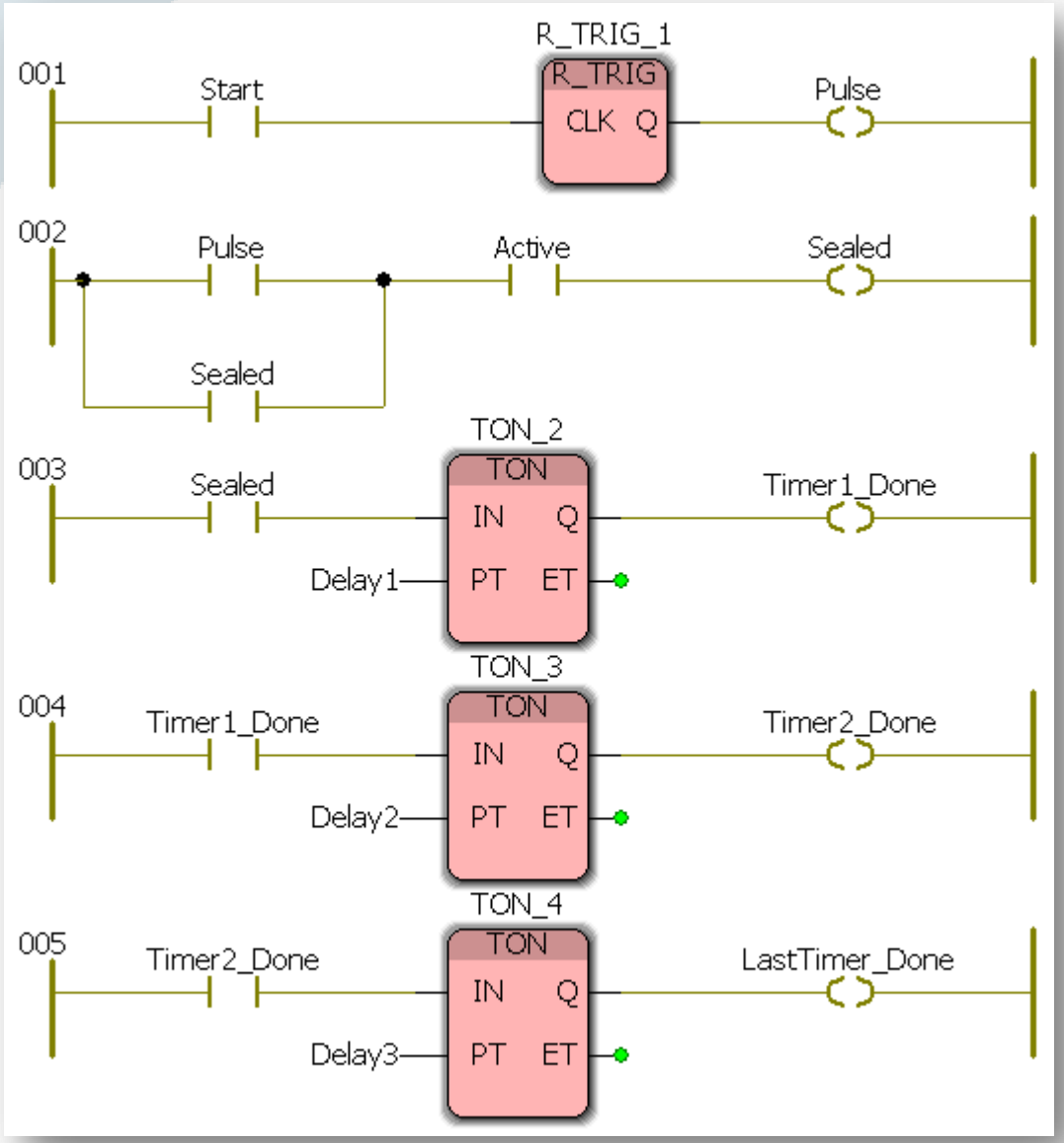


Open block properties.
"Negate" the input



Timer Sequencing

- *Use ladder "seal-in" circuit*



Arrays

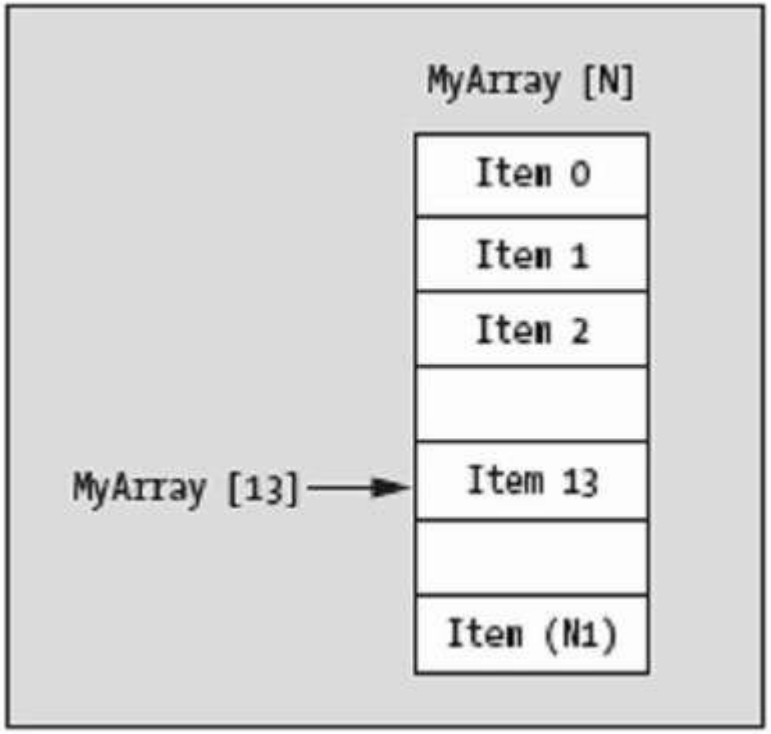
Definition
Create Data Type
Usage and Syntax

YASKAWA

What is an Array?

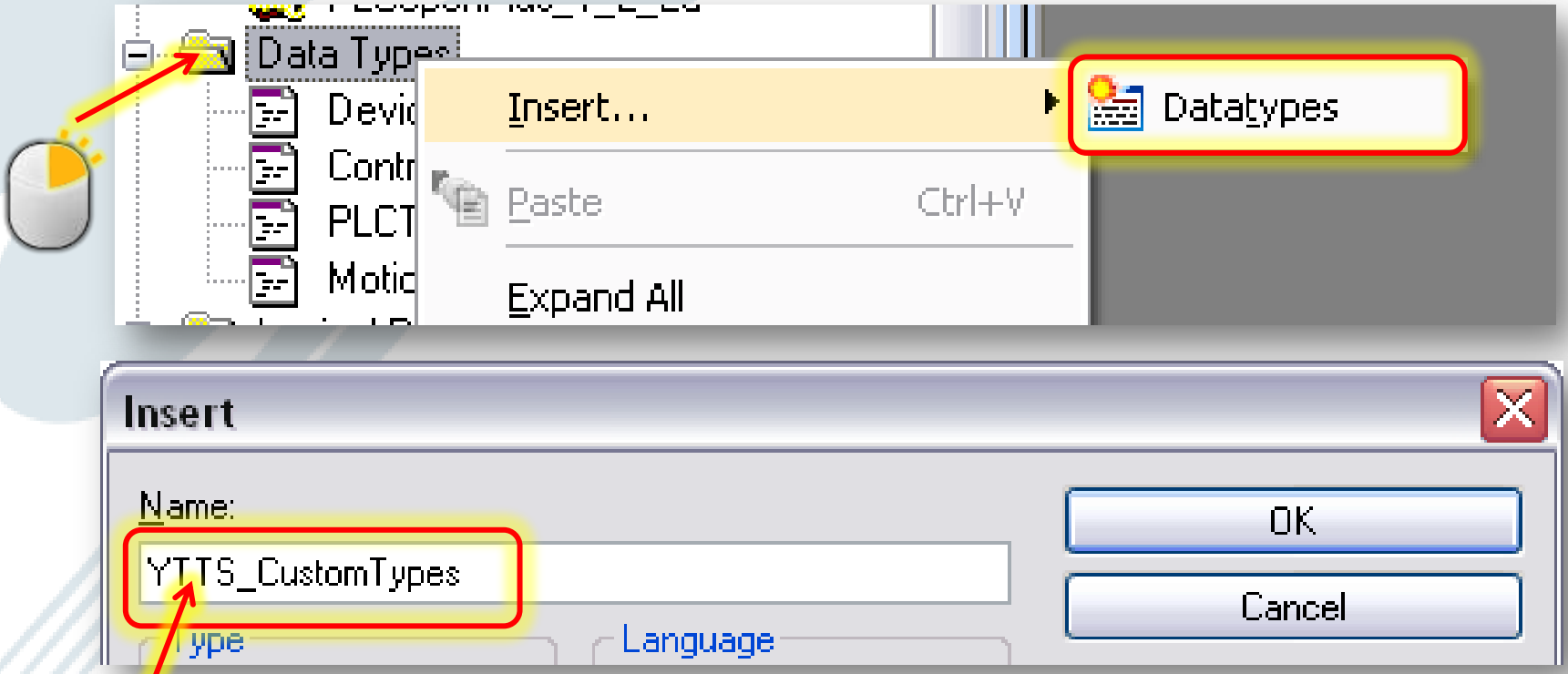
- *A Single Variable*
- *Multiple elements*
- *Same data type*
- *Values accessed using index*

- *Using an Array*
  - *Create array data type*
  - *Create a variable with this data type*
  - *Array element index*
    - » *MyArray[13]*



Data Type Worksheet

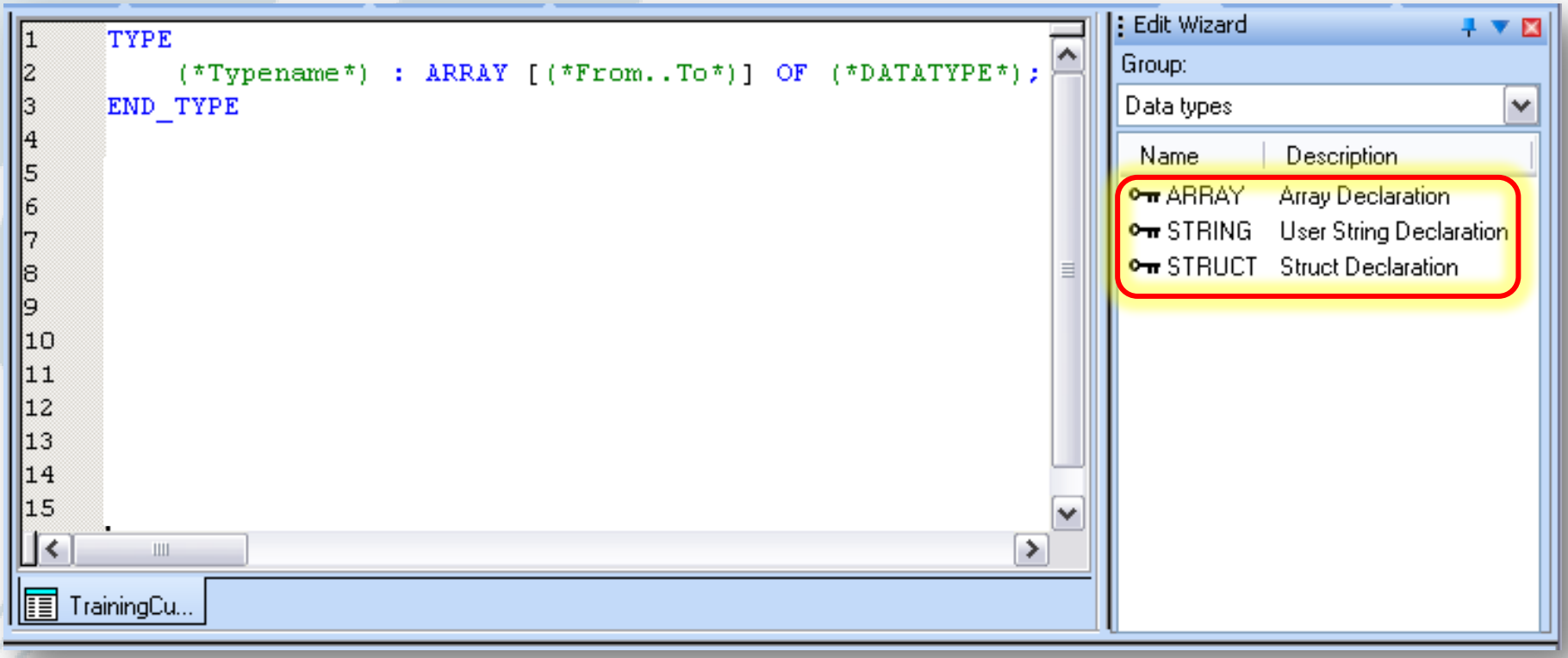
- *Arrays and Structures Defined as Data Type*
  - *Derived data types*



Best Practice: Prefix data type name
1. Avoid name conflict
2. Organization

ARRAY Data Type

- *Add ARRAY Data Type From Edit Wizard*



Create Array Data Type

- *Compile (Make)*

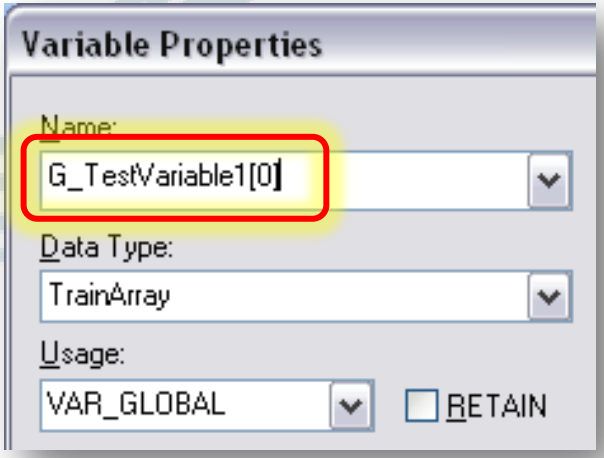
```
TYPE
     YTTS_TrainArray : ARRAY [0..7] OF WORD;
END_TYPE
```

Create Variable

- *Create Variable With Data Type*

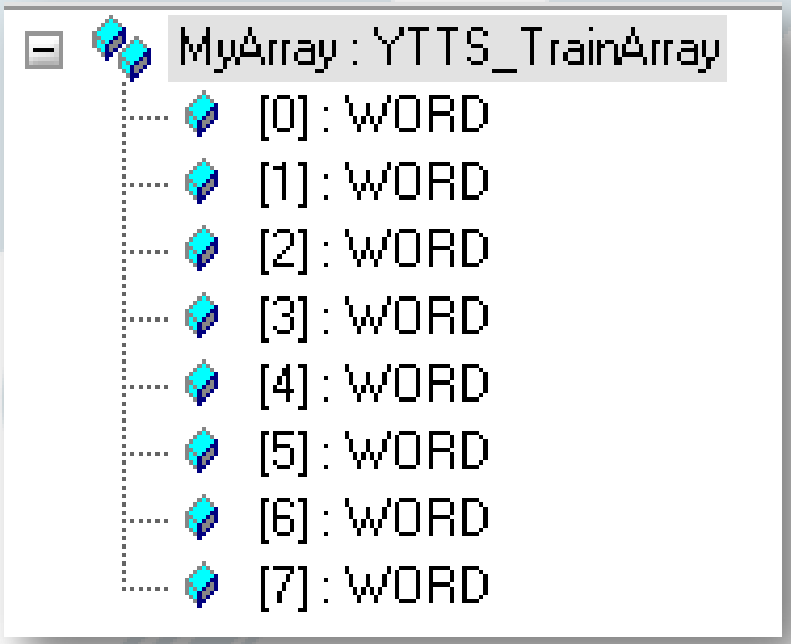
	Name	Type	Usage
	⊞ System		
	⊟ User Variables		
	G_TestVariable1	TrainArray	VAR_GLOBAL
	G_TestVariable2	TrainStruct	VAR_GLOBAL

- *Use Variable In Code*
  - *Array Element [ ]*

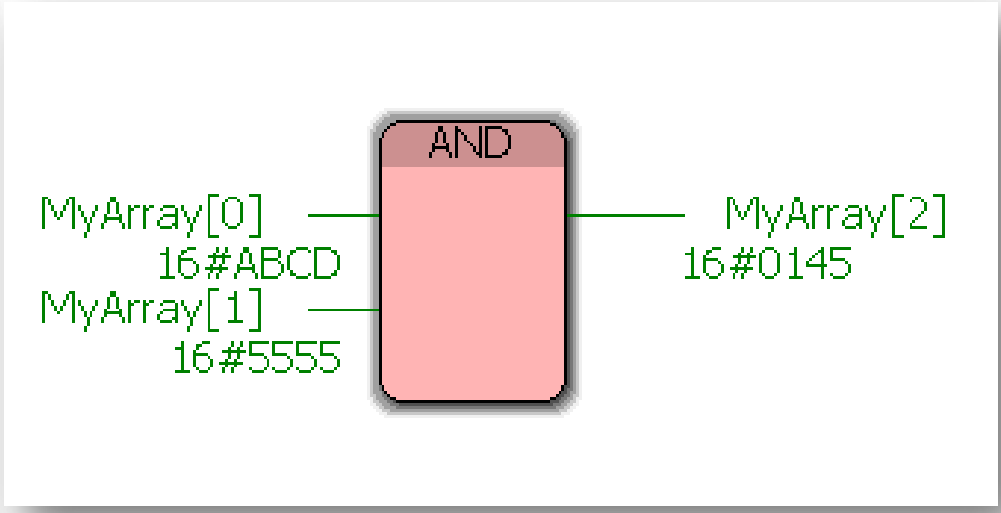


Array in Code

- *Add to Watch Window*



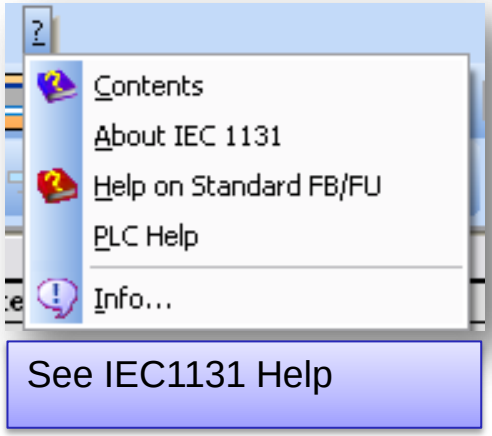
- *Use in Code*

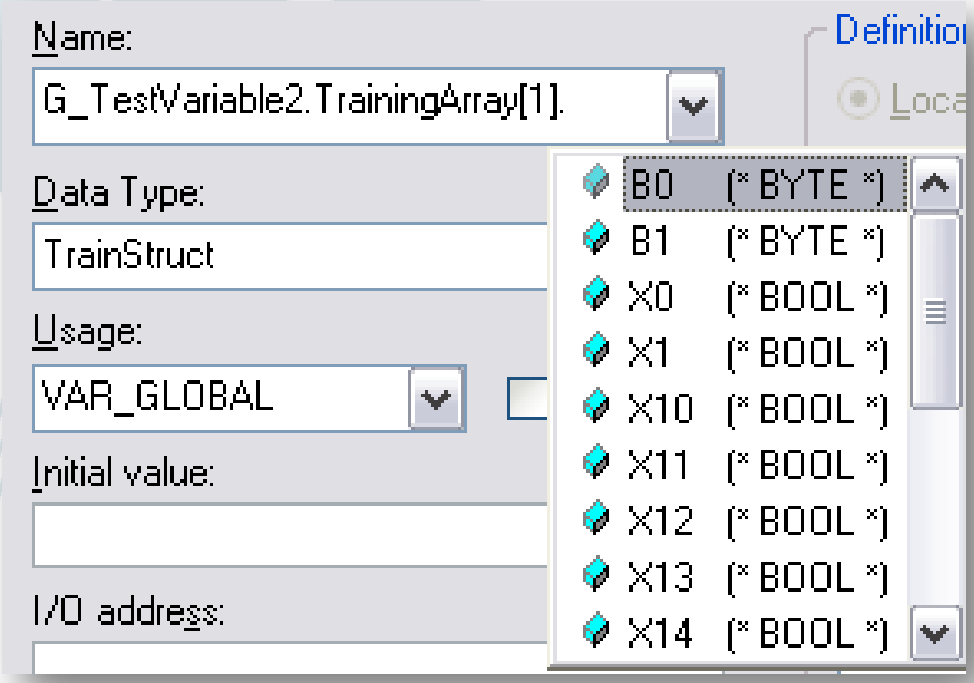


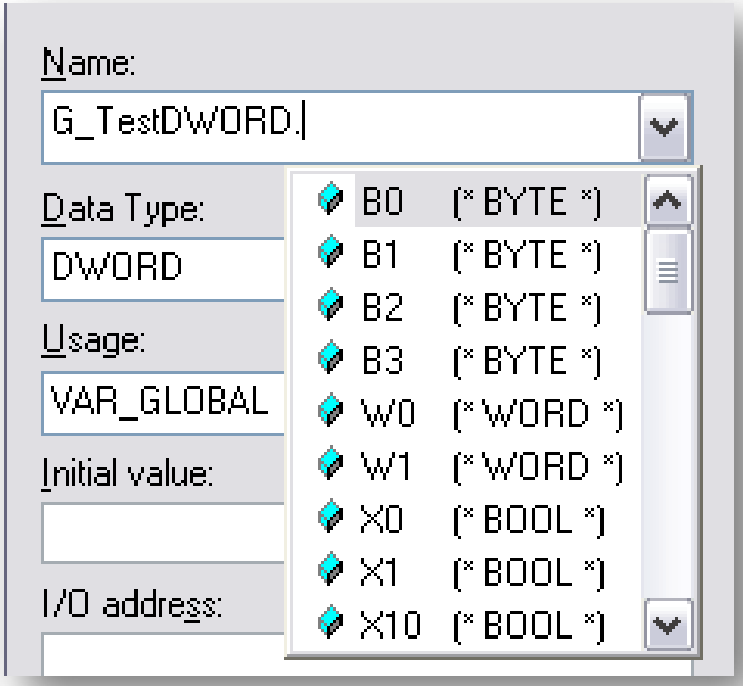

```
n:= 0;                          (* n is an integer *)
MyArray[n]:= Var1 ;             (* apply value to array element *)
n:= n+ INT#1;                   (* increment index variable *)
```

BYTE, WORD, DWORD

- *"Bit String"*
  - *BYTE, WORD, DWORD*
  - *Default Debug Display: Hexadecimal (0-F)*
  - *Use "dot" (.) to specify bit, byte, etc*







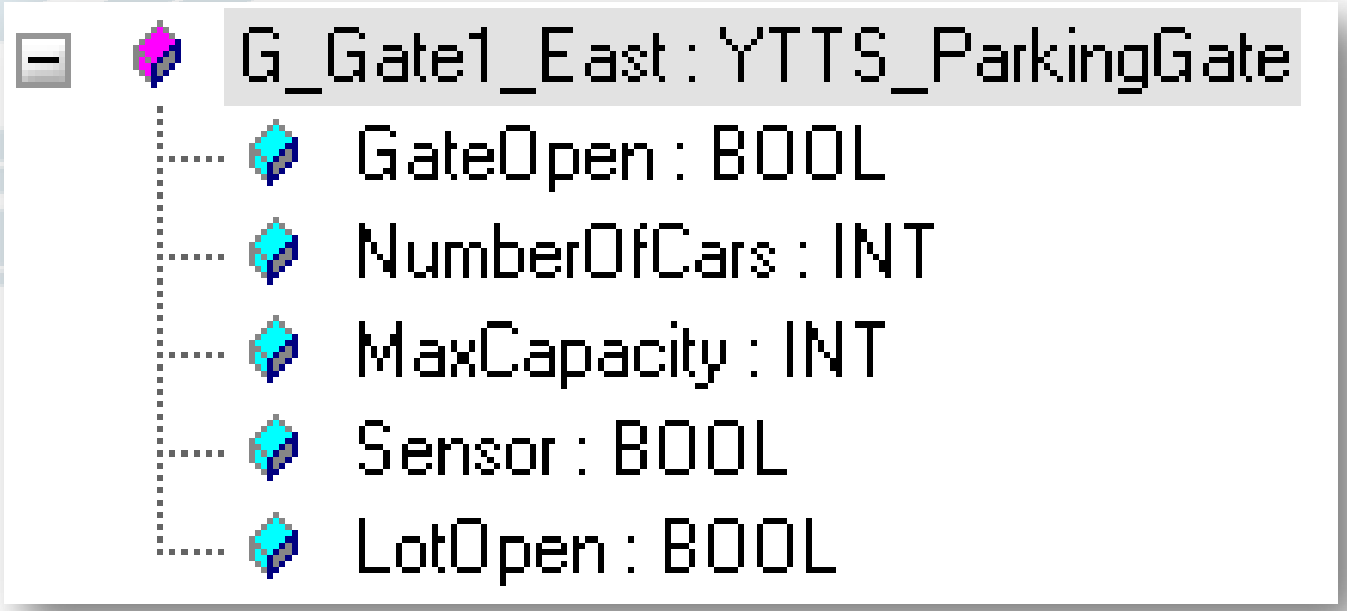
Structures

Definition
Create Data Type
Usage and Syntax

YASKAWA

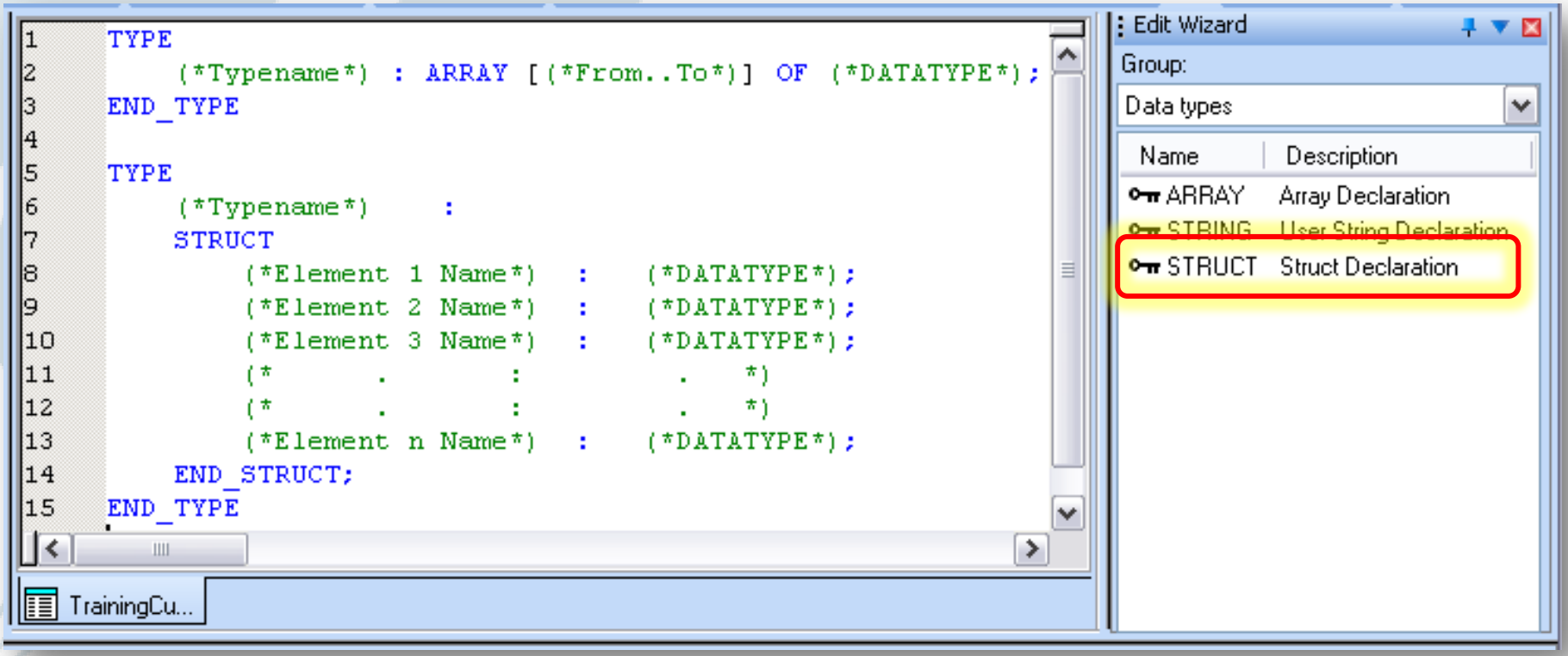
What is a structure?

- *A set of data with mixed data types*
- *Similar to a file directory structure*
- *Organize Data*
- *Complex names (DOT notation)*
  - *G_Gate1_East.Sensor*



STRUCT Data Type

- *Add STRUCT Data Type From Edit Wizard*



Create STRUCT Datatype

- *Compile (Make)*

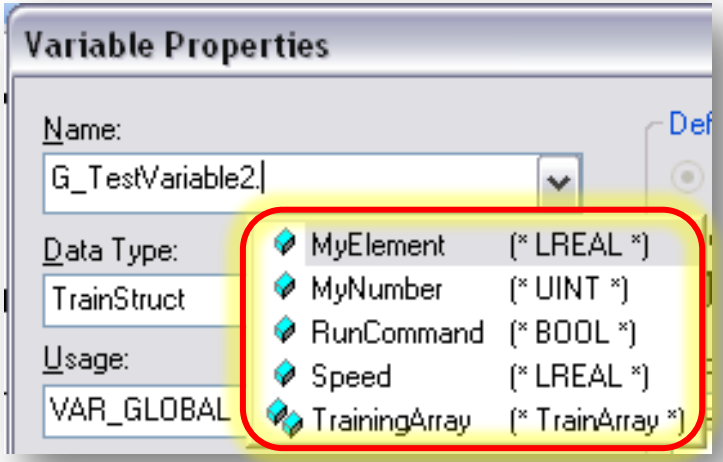
```
TYPE
    YTTS_TrainArray : ARRAY [0..7] OF WORD;
END_TYPE
TYPE
    YTTS_TrainStruct    :
    STRUCT
        Array8   :   YTTS_TrainArray;
        Frequency   :   LREAL;
        RunCommand   :   BOOL;
        Parameter   :   UINT;
    END_STRUCT;
END_TYPE
```

Structure/Array In Code

- *Create Variable In List*

	Name	Type	Usage
	⊞ System		
	⊟ User Variables		
	G_TestVariable1	TrainArray	VAR_GLOBAL
	G_TestVariable2	TrainStruct	VAR_GLOBAL

- *Use Variable In Code*



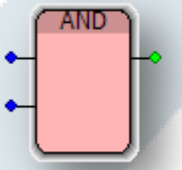
Driving Value

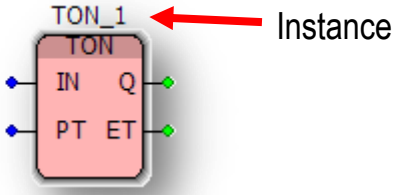
Functions & Function Blocks

Comparison
Bitwise Functions
Duplicate & Negate Inputs
EN/ENO

YASKAWA

- *Function Block Definition (IEC 61131-3)*



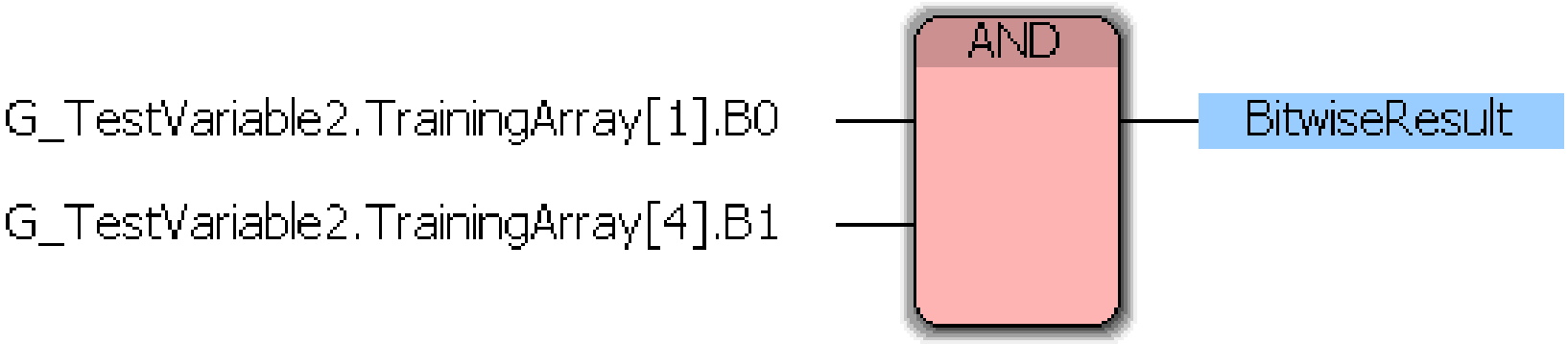


Function (FU)	Function Block (FB)
Executes in 1 scan	May require several scans to execute.
Executes without an "instance" of memory, as indicated by the icon	Each FB runs in a separate "instance" of memory, as indicated by the icon
Single Output required	Multiple Outputs possible
Must connect output and **all** inputs for compile	Only Var_In_Out must be connected to compile
Output value depends only on the function inputs	Output value may depend on values that are not part of the function block input
Function Blocks may not be used inside functions*	Other function blocks may be used within the function block*
Most efficient use of processor and RAM*	May use slightly more processor power and RAM to accomplish the same task*
VAR (local) variables only*	VAR_GLOBAL allowed, but not recommended*

* For user created FU or FB

Bitwise Functions

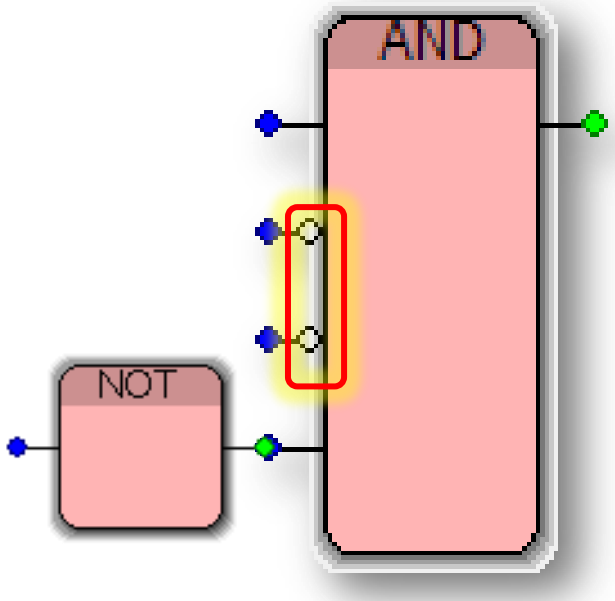
- *Functions operate at bit level*
- *Inputs require same data type*

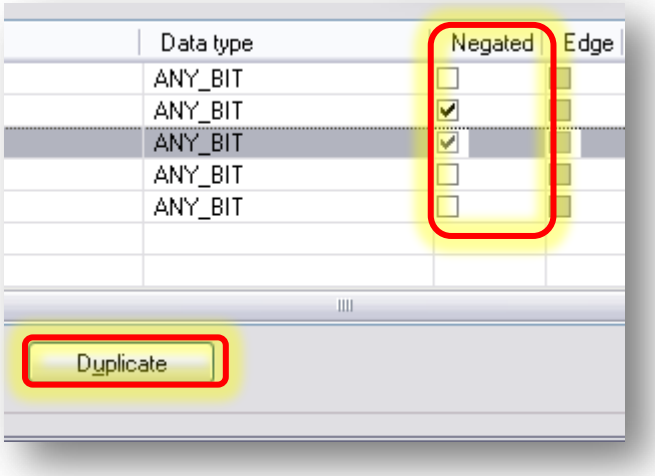


	Bit (X)							
Data	X7	X6	X5	X4	X3	X2	X1	X0
Input 1	1	0	0	0	1	0	0	0
Input 2	0	0	1	0	1	0	0	1
Input n								
Output	0	0	0	0	1	0	0	0

Properties

- *Negate*
  - *Same as NOT function*
  - *Inverts ALL bits*
- *Duplicate*
  - *Add Inputs*
  - *Does not apply to all Functions*

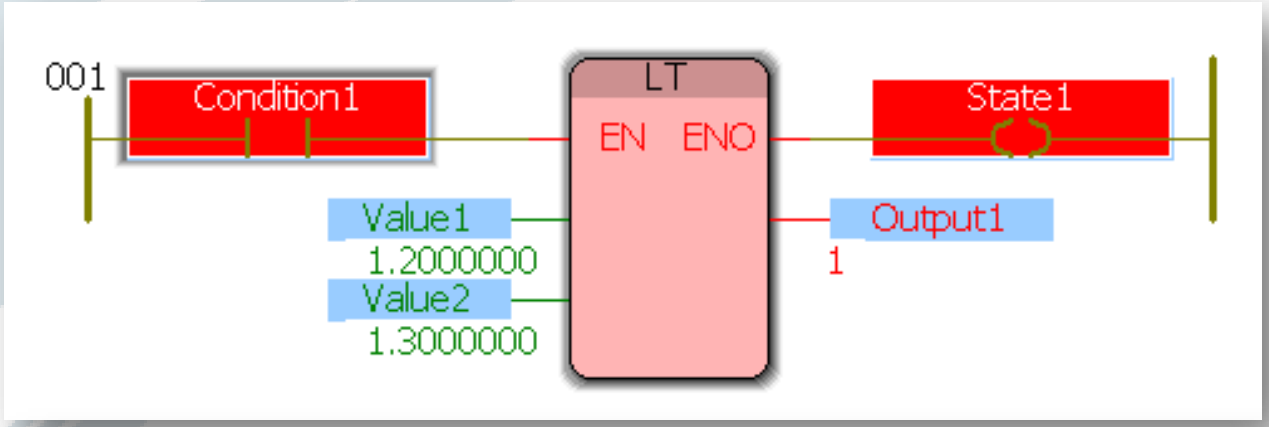




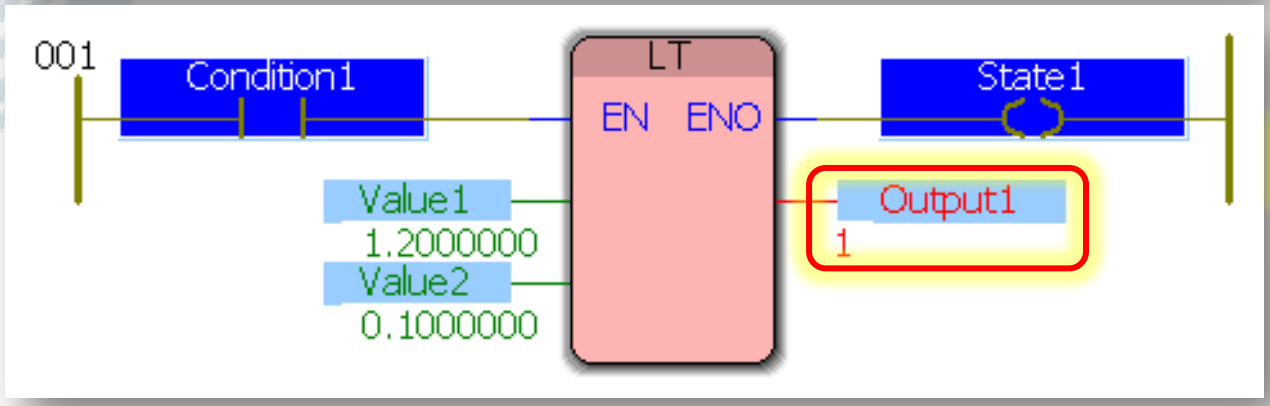
EN / ENO

- *Definition: Enable / Enable Output*
  - *Enable or disable execution*
  - *EN disconnect = Always enable execution*
  - *Normal operation ➔ EN = ENO*
  - *Function error ➔ EN = ON, ENO=OFF*

See Yaskawa FAQ number
MTN-7PCQK4



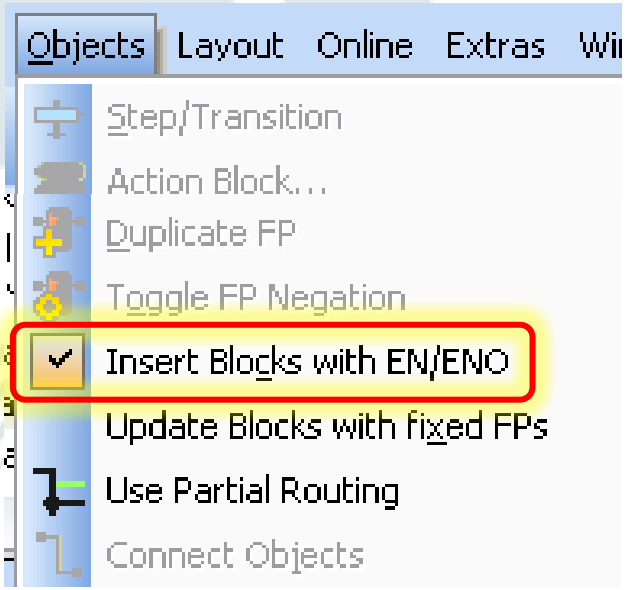
Function Enabled
Execution = TRUE
Output updates every scan



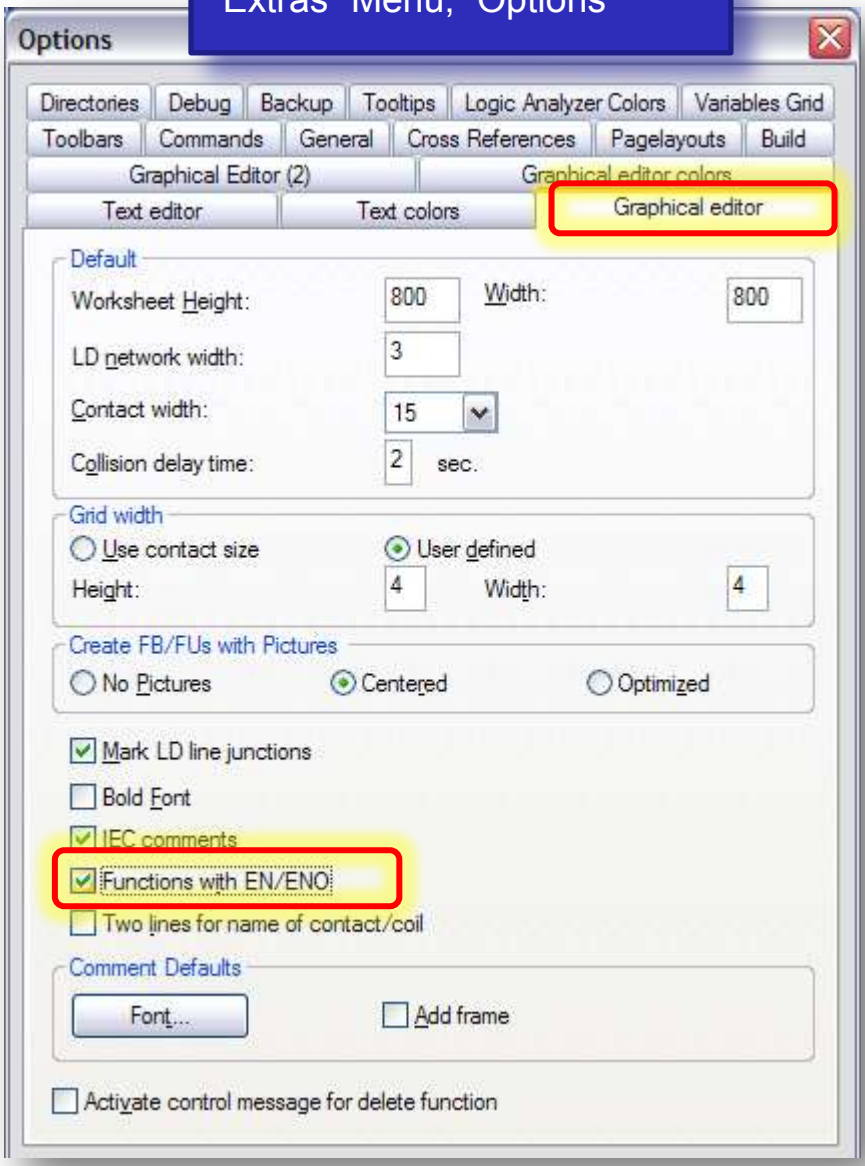
Function Disabled
Execution = FALSE
Output remains at previous state

EN / ENO

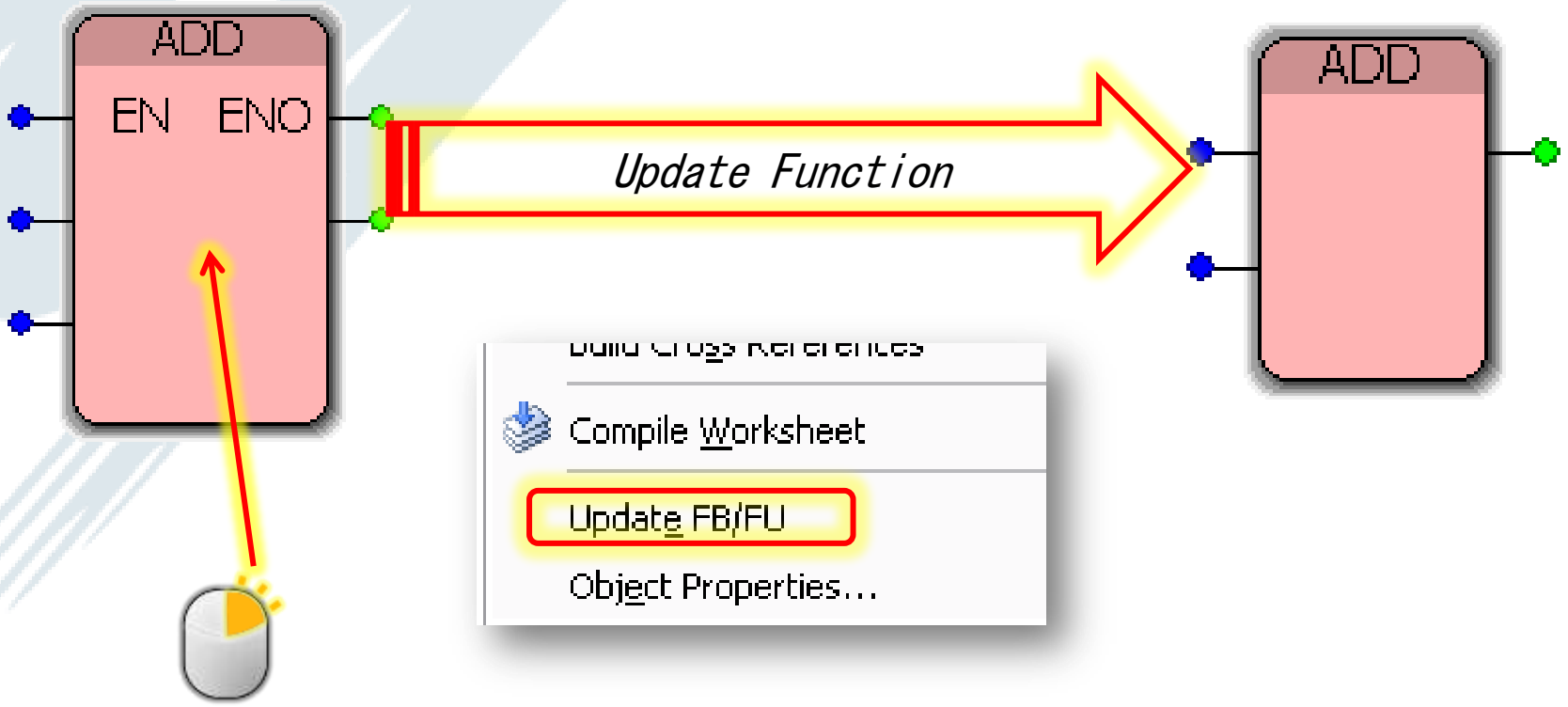
- *Software Option*
  - *Applies to all worksheets and new worksheets*
  - *Affects new functions only*
- *Worksheet Option*
  - *Active worksheet only*
  - *Affects new functions only*



"Extras" Menu, "Options"



- *Manual Update of EN/ENO For Existing Functions*



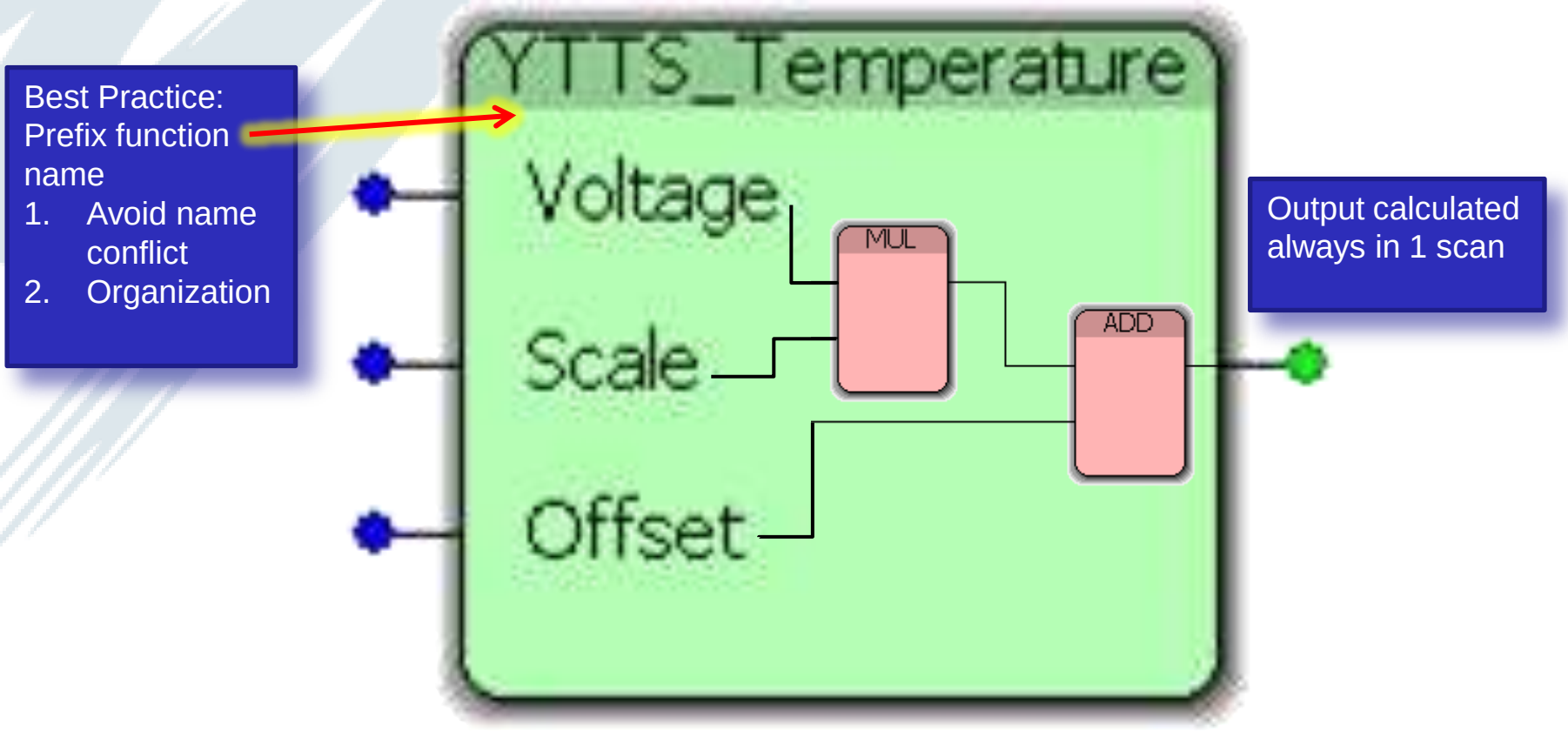
Driving Value

User Function

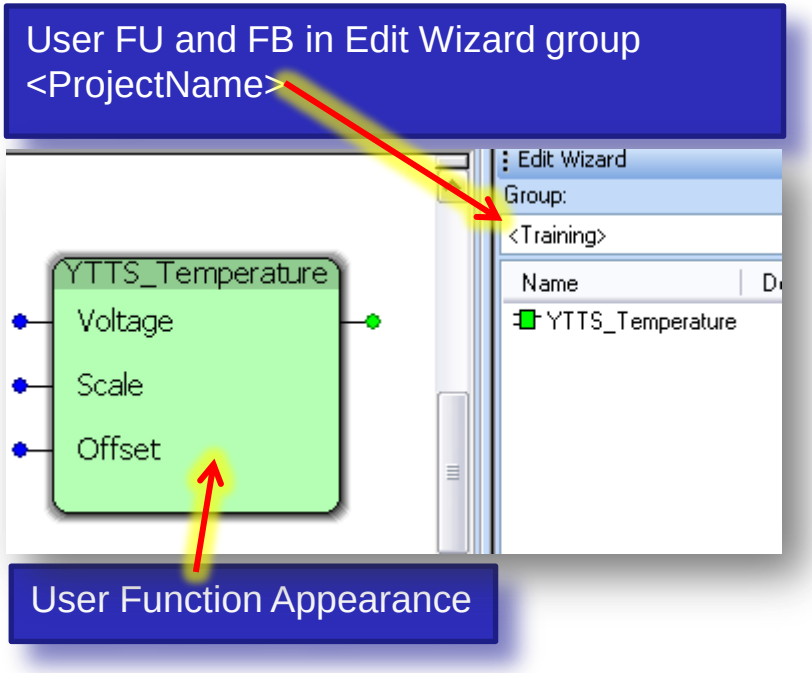
Overview & Concept
Write Code
Function in Program POU
PowerFlow
Function Troubleshooting

YASKAWA

- *YTTS_Temperature*
  - *Convert analog input voltage to temperature*
    - » *Based on scale and offset*
    - » *Linear equation: Y=mX+b*

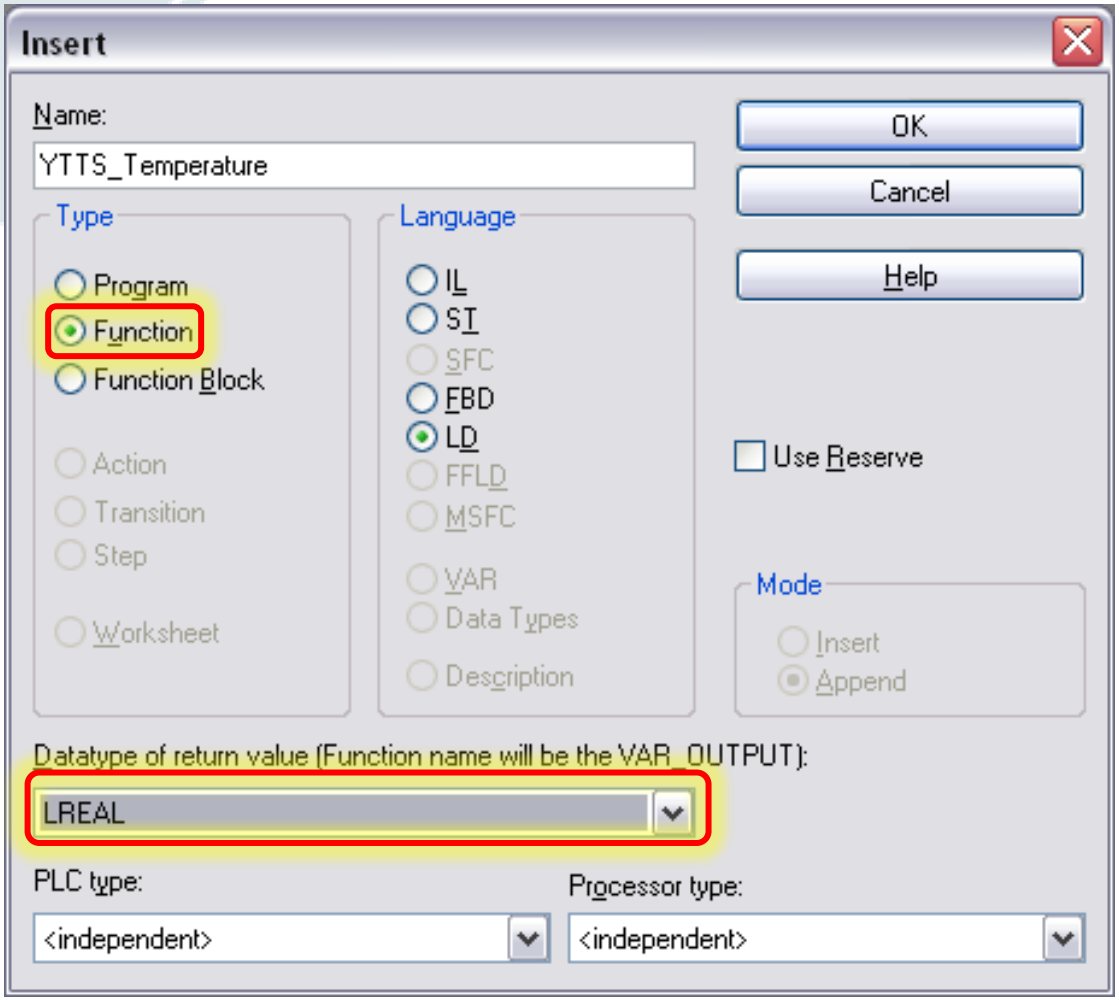


- *Write a User Function POU*
  - *Convert analog input voltage to temperature*
    - » *Based on scale and offset*
    - » *Linear equation: Y=mX+b*
  - *Temperature*
    - » *output of function*
  - *Scale*
    - » *input "slope" of function*
  - *Voltage*
    - » *raw input voltage*
  - *Offset*
    - » *calibration offset*



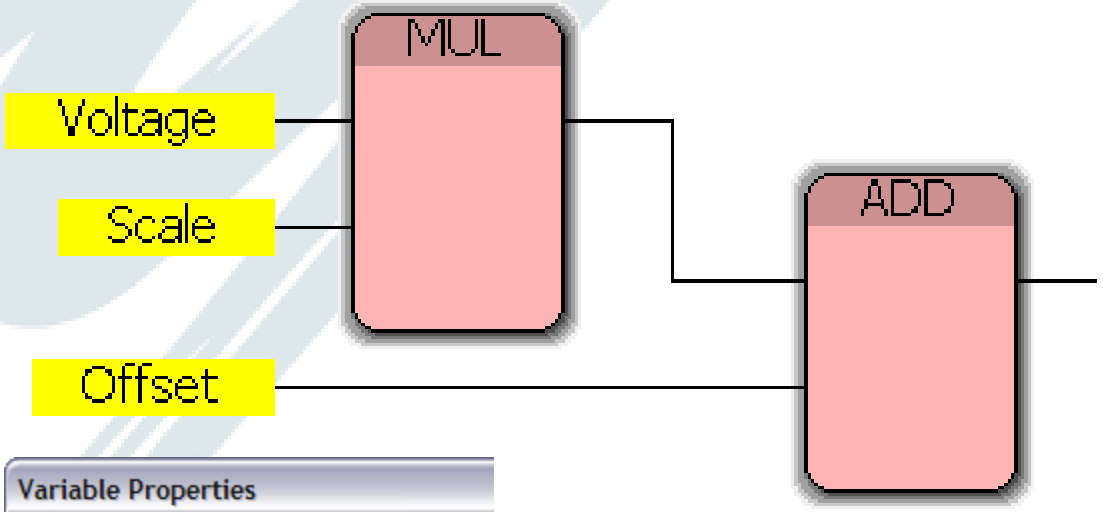
Insert Function

- *New Function POU*
  - *Name:*
    - » *YTTS_Temperature*
  - *Type:*
    - » *Function*
  - *Language:*
    - » *LD*
  - *Datatype of return value:*
    - » *LREAL*

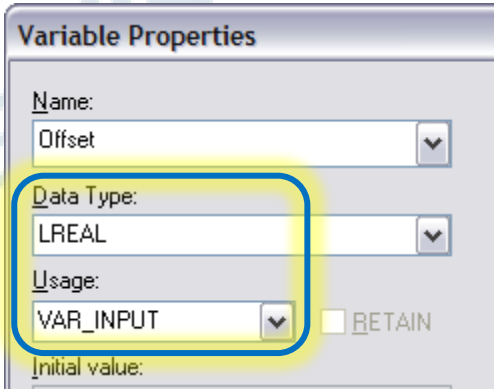


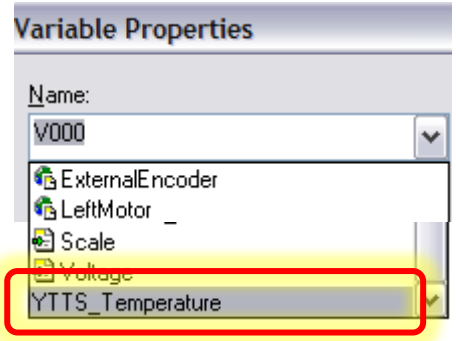
Function Code

- *Inputs*
  - *Usage: VAR_INPUT*
  - *Datatype: LREAL*
- *Output*
  - *Choose function name*

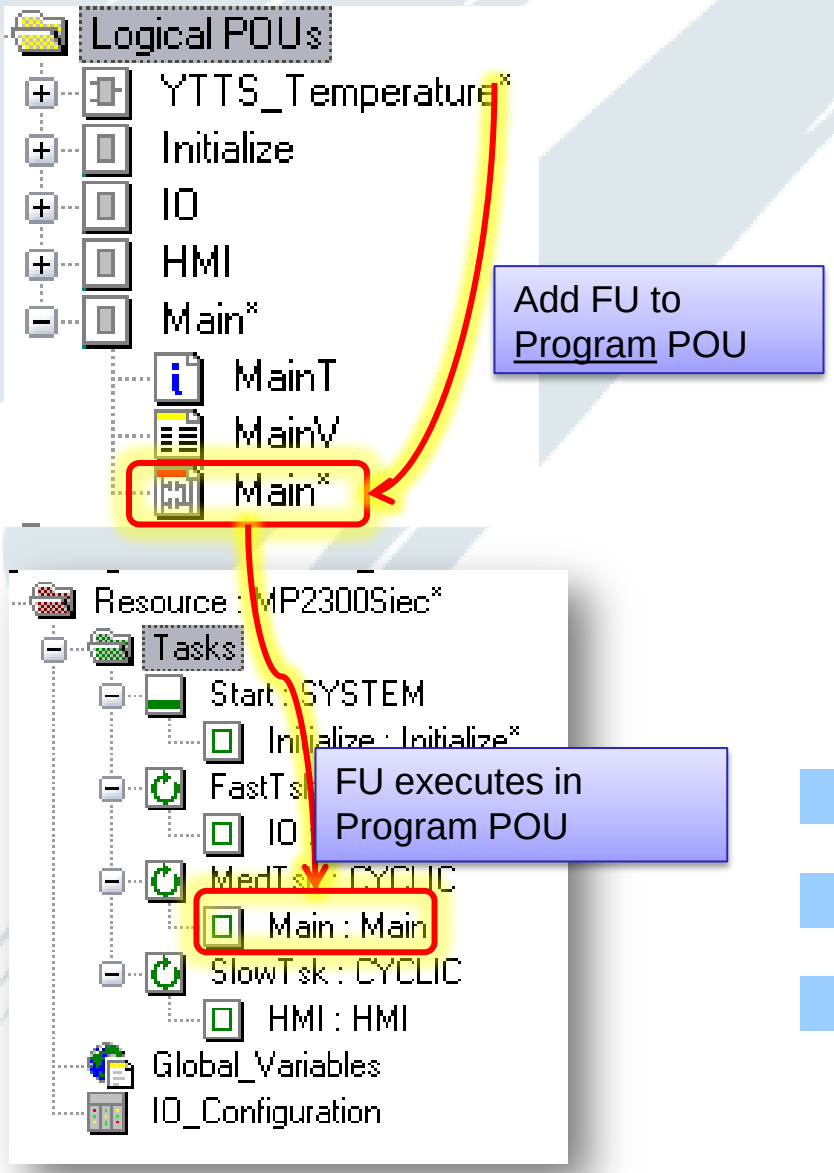


Another variable may not use this name!

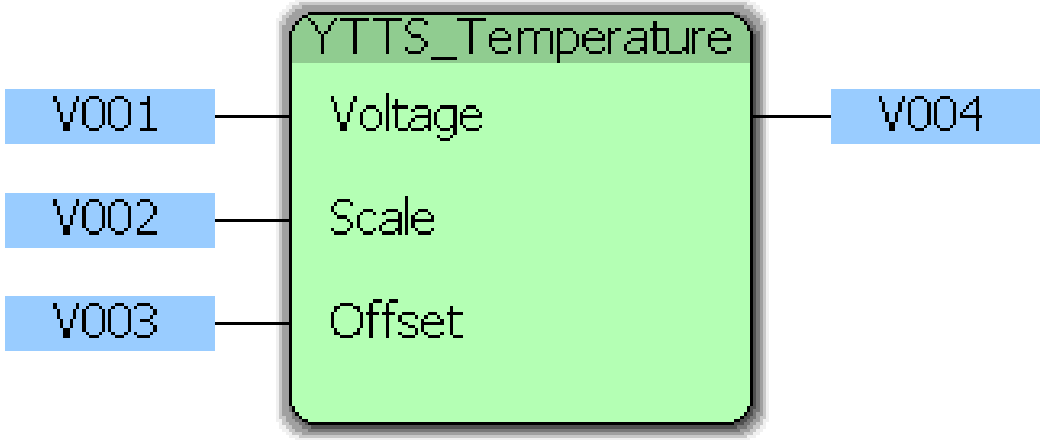




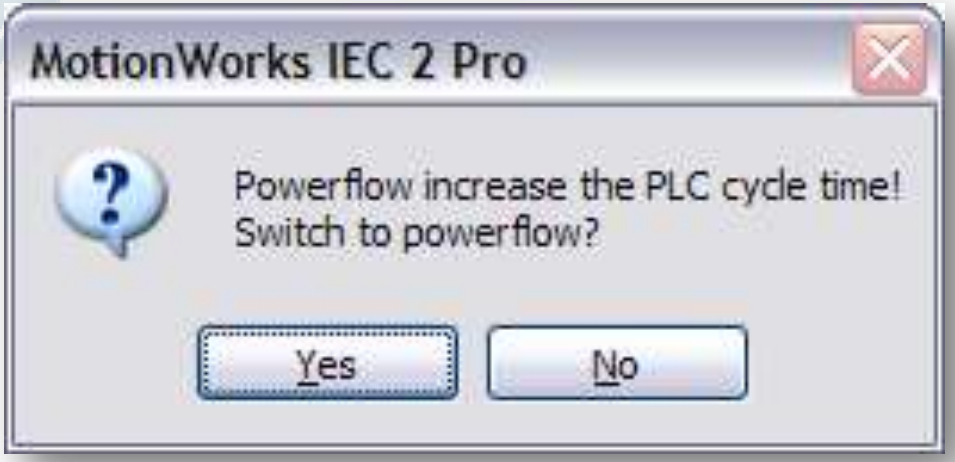
Function Execution



- *MAKE*
- *Execute YTTS_Temperature in Main Program POU*
  - *Open logical POU "Main"*
  - *Add Function*
    - » *Edit wizard group <project name>*
  - *Connect variables*
  - *Download Changes*
  - *Confirm output*

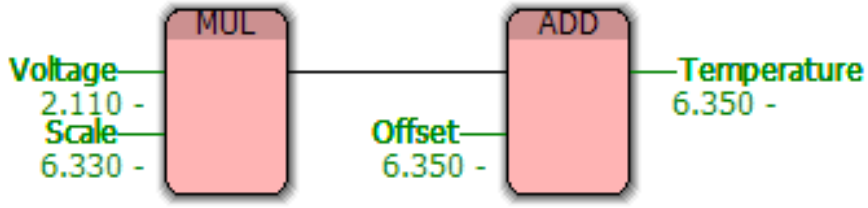


- *PowerFlow Prompt*
  - *Debug feature*
  - *Condition: function worksheet open*
  - *ST language support*
    - » *monitor lines executed*



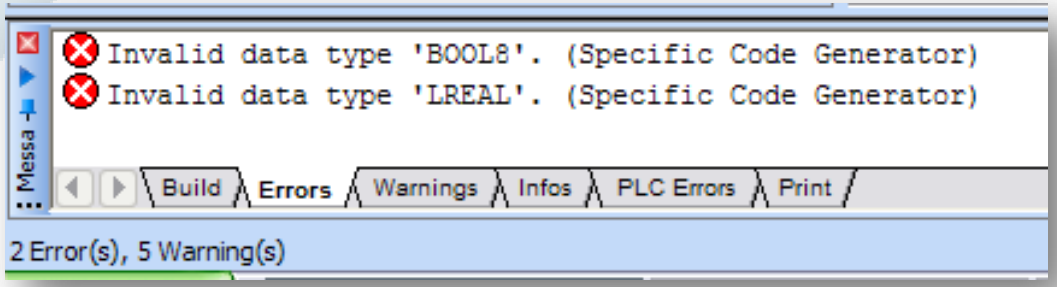
Click **No** to disable this feature

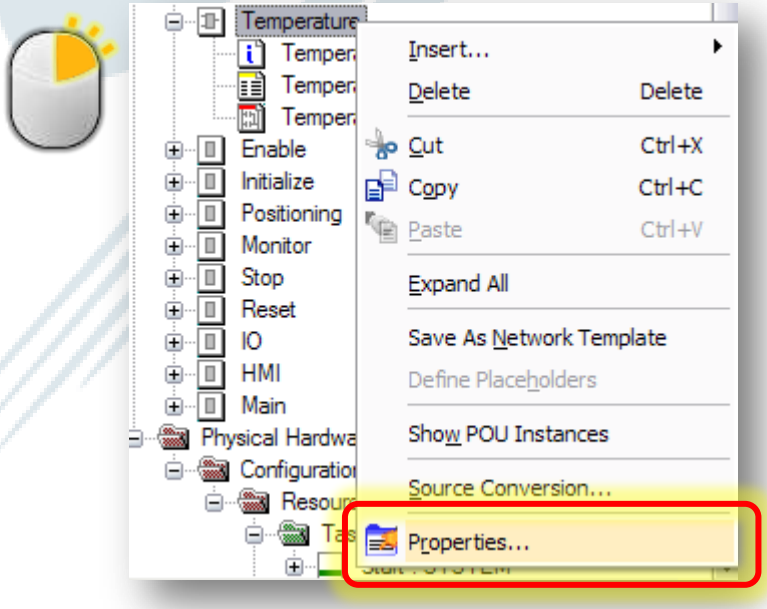
- *LD , FBD not supported*
  - » *Incorrect display*
  - » *Answer "No"*

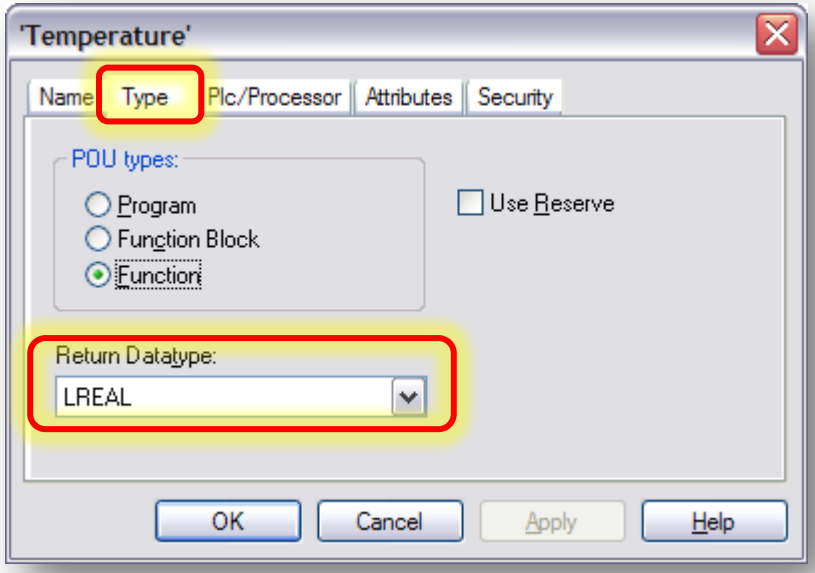


Function Properties

- *Return Datatype*
  - *Return Datatype may be incorrect*



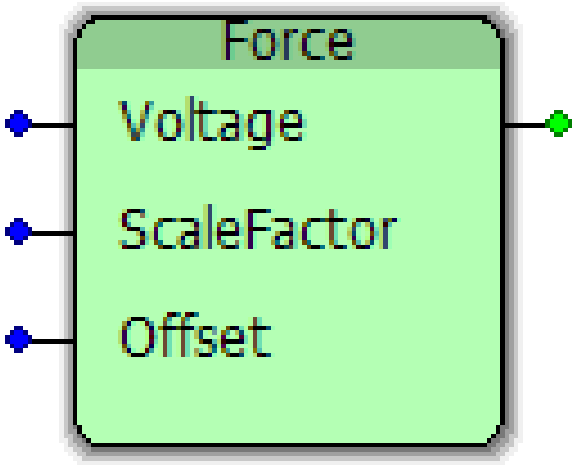




Create the same function using ST language

- *Force = Voltage * ScaleFactor + Offset*

HINTS:
1. This can be done in 1 line of ST
2. Use F5 to declare variables in ST
3. Refer to the Initialize POU for syntax;



Driving Value

User Function Block

Overview & Concept
Write Code
Implement Instance
Troubleshooting
Enable Input
Customized Help

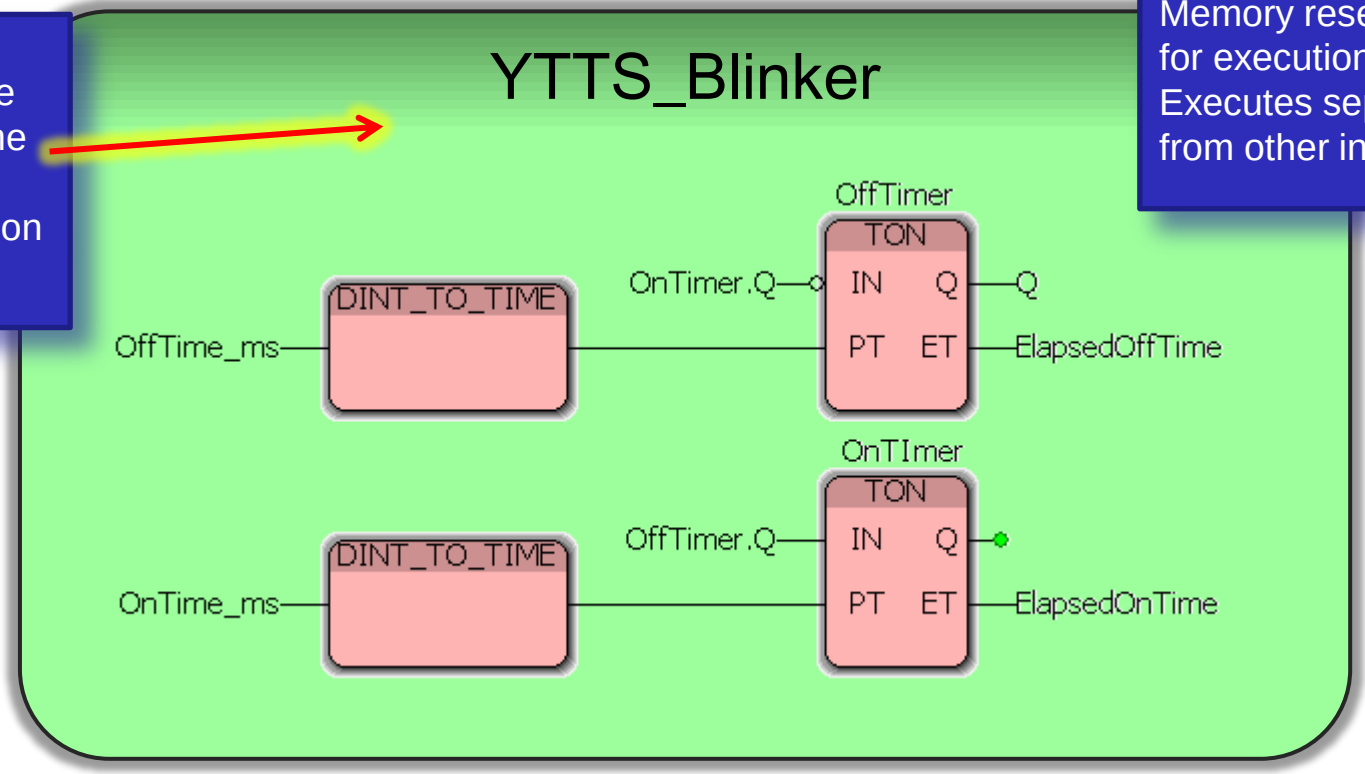
YASKAWA

- *YTTS_Blinker*
  - *Run in program POU*

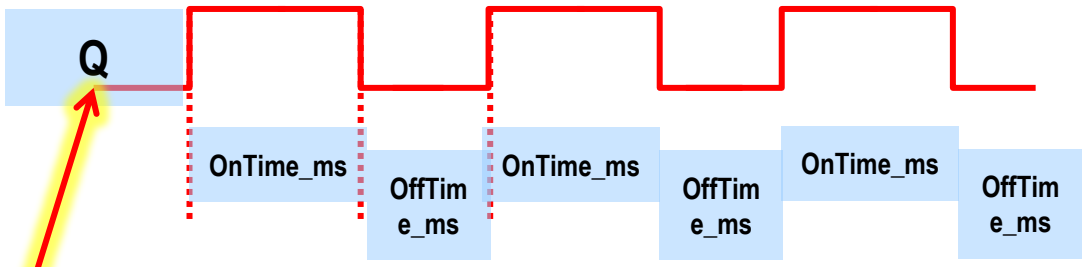
Outputs calculate over many scans. Instance required. Memory reserved for execution. Executes separately from other instances

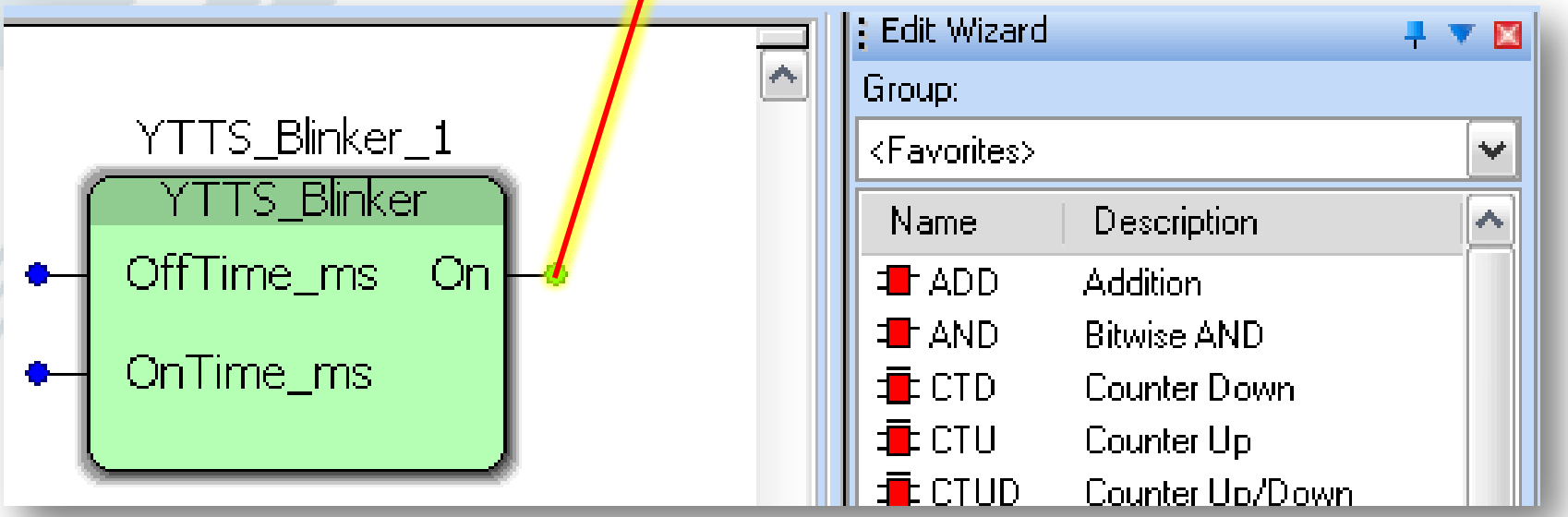
•*Main POU*

Best Practice:
Prefix FB name
1. Avoid name conflict
2. Organization



- *Write a User Function Block POU*
  - *"Q" output cycles according to input OnTime and OffTime*
  - *Inputs*
    - » *OffTime_ms*
    - » *OnTime_ms*
  - *Outputs*
    - » *Q*





Write Function Block Code

HINT: Check the Type, Usage, Init and Retain of each variable

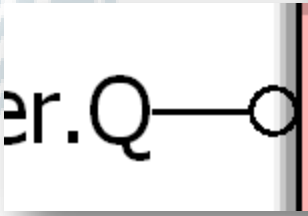
Best Practice: No Global Variables in a Function Block

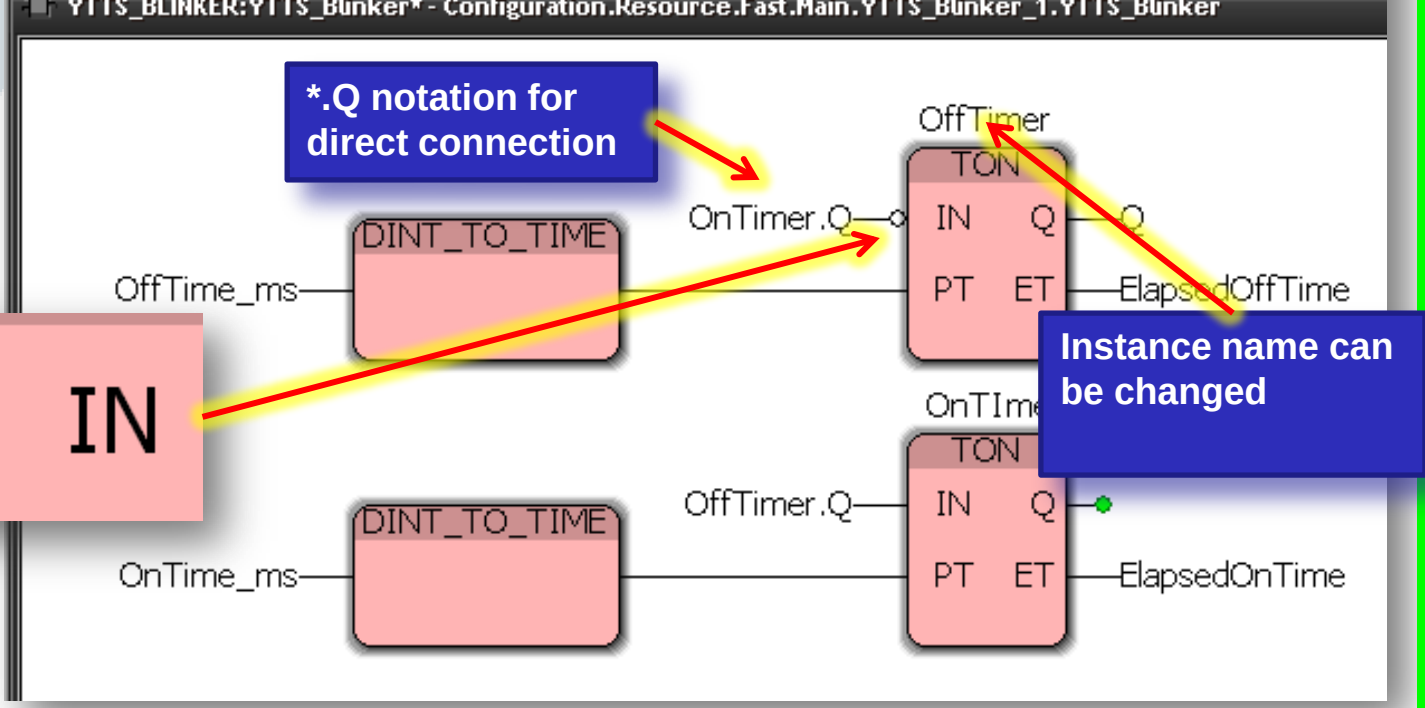
YTTS_BlinkerV:YTTS_Blinker

Name	Type	Usage	Init	Description	Address	Retain
Default						
OffTimer	TON	VAR				
OnTimer	TON	VAR				
ElapsedOffTime	TIME	VAR				
ElapsedOnTime	TIME	VAR				
OffTime_ms	DINT	VAR_INPUT	DINT#1000			
OnTime_ms	DINT	VAR_INPUT	DINT#1000			
Q	BOOL	VAR_OUTPUT				

YTTS_BLINKER:YTTS_Blinker* - Configuration.Resource.Fast.Main.YTTS_Blinker_1.YTTS_Blinker

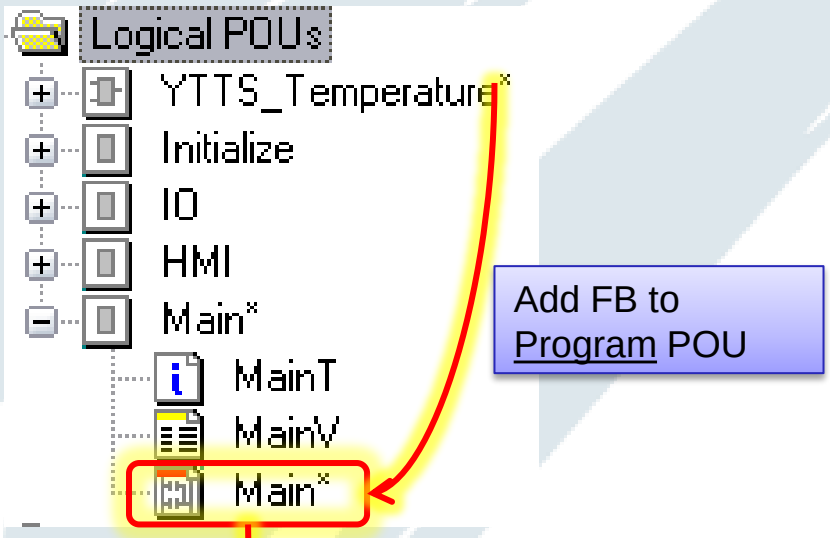
***.Q notation for direct connection**

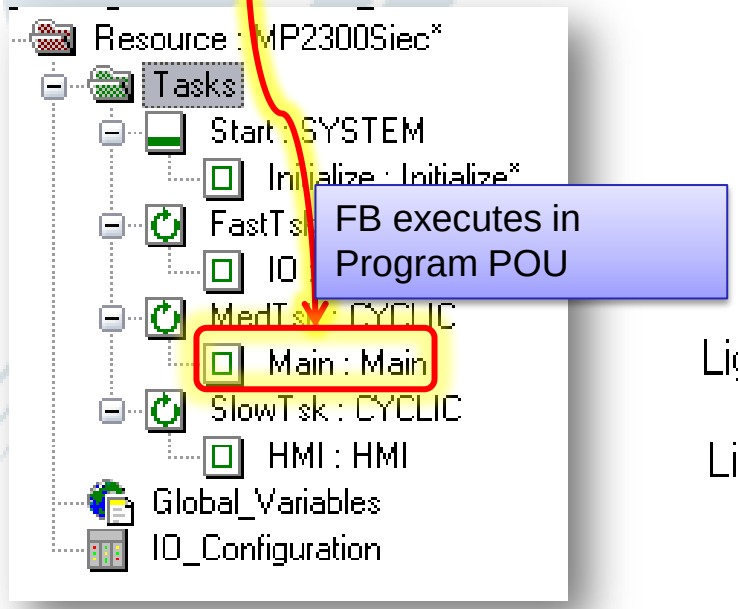




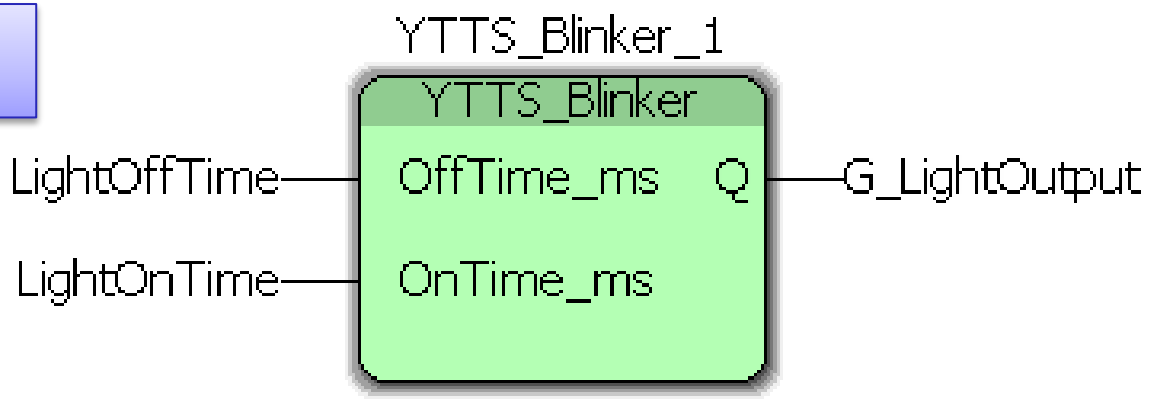
Instance name can be changed

Function Block Execution



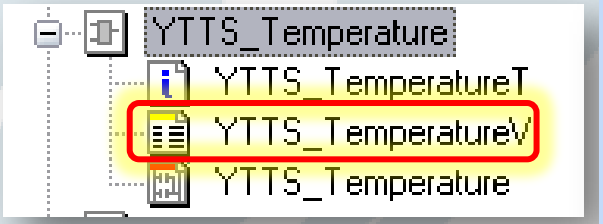


- *Execute YTTS_Blinker*
  - *Add to Main Program POU*
  - *Connect variables*
  - *Confirm output*
- *Extra Credit*
  - *Drive simulator output*
  - *Use best practice for IO*

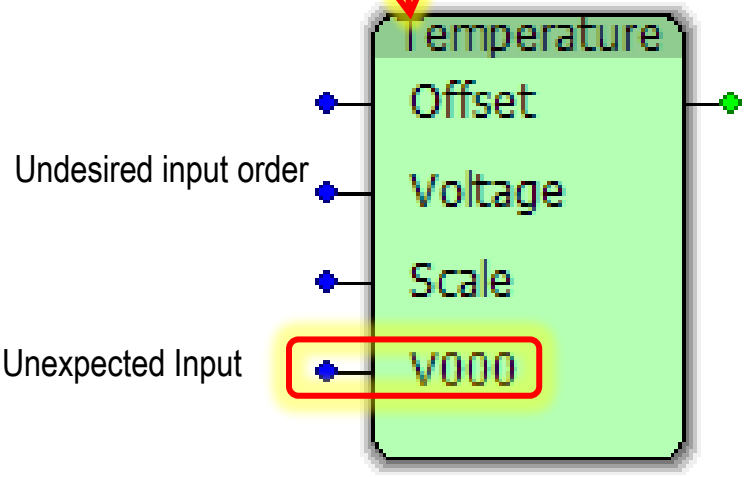


Troubleshooting - Input Order

- *Local Variable List*
  - *Defines Inputs and appearance*



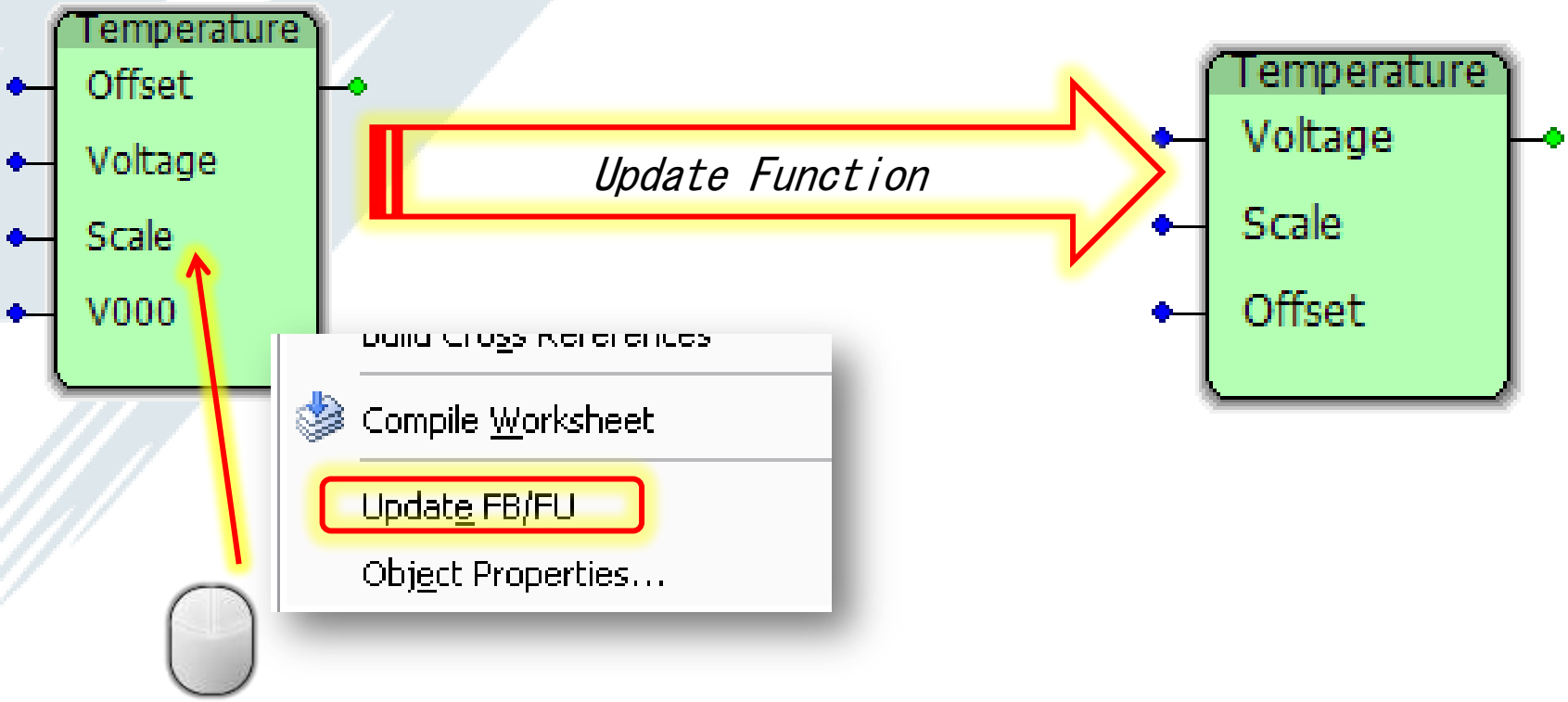
	Name	Type	Usage	Description
	Default			
	Offset	LREAL	VAR_INPUT	Temperature offset for calibration
	Voltage	LREAL	VAR_INPUT	Raw input voltage to controller
	Scale	LREAL	VAR_INPUT	Scale factor to convert volts to temperature
	V000	BOOL	VAR_INPUT	ooops! I didn't mean to add this variable!



Order in list = Order of appearance

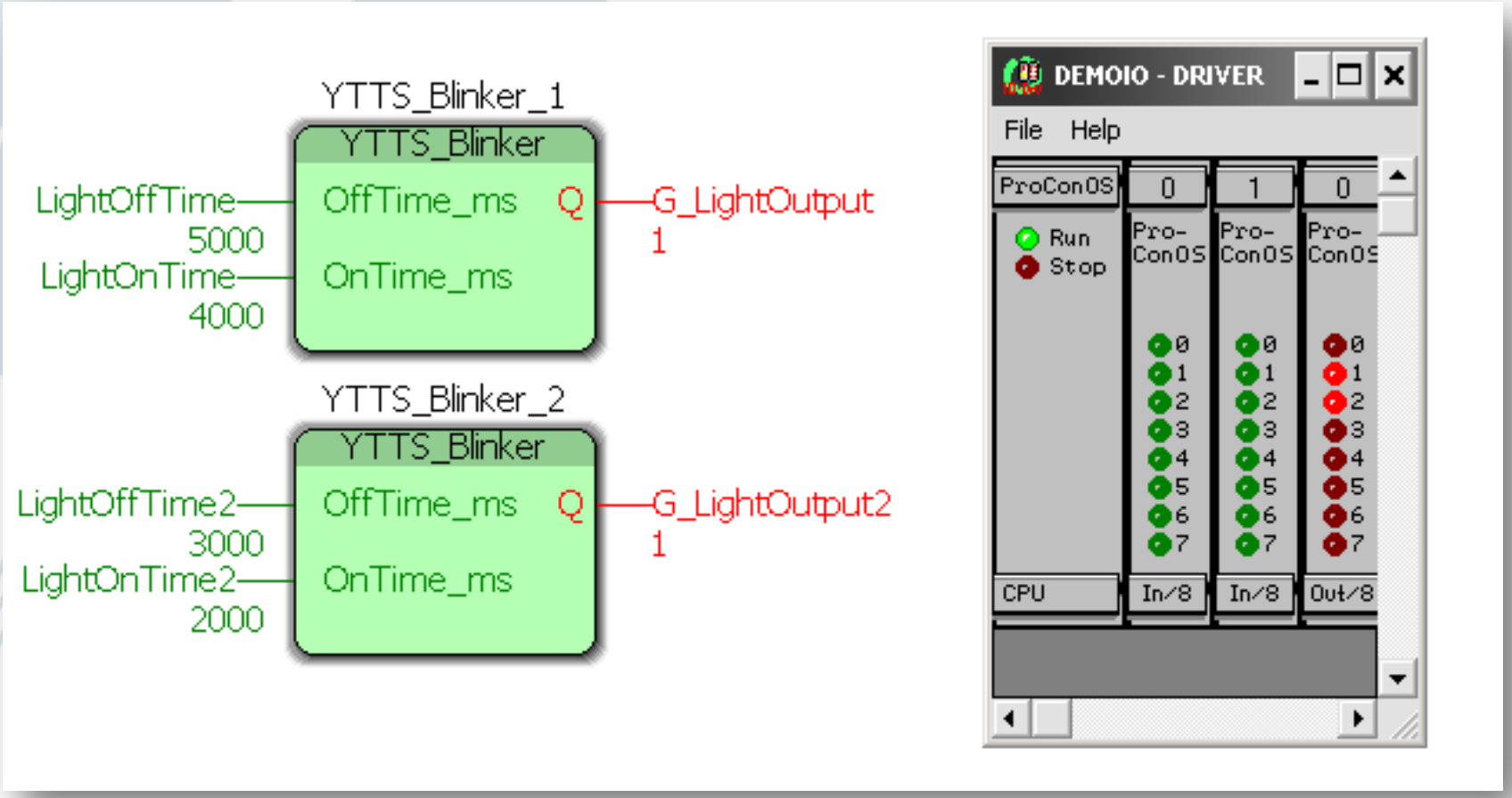
Troubleshooting - Update

- *Make Project*
- *Right-Click*
  - *Update FB/FU*



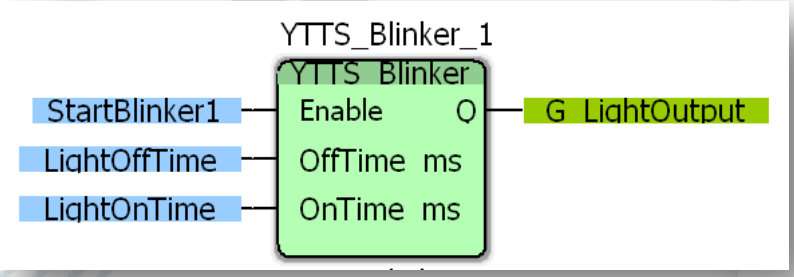
Function Block Instances

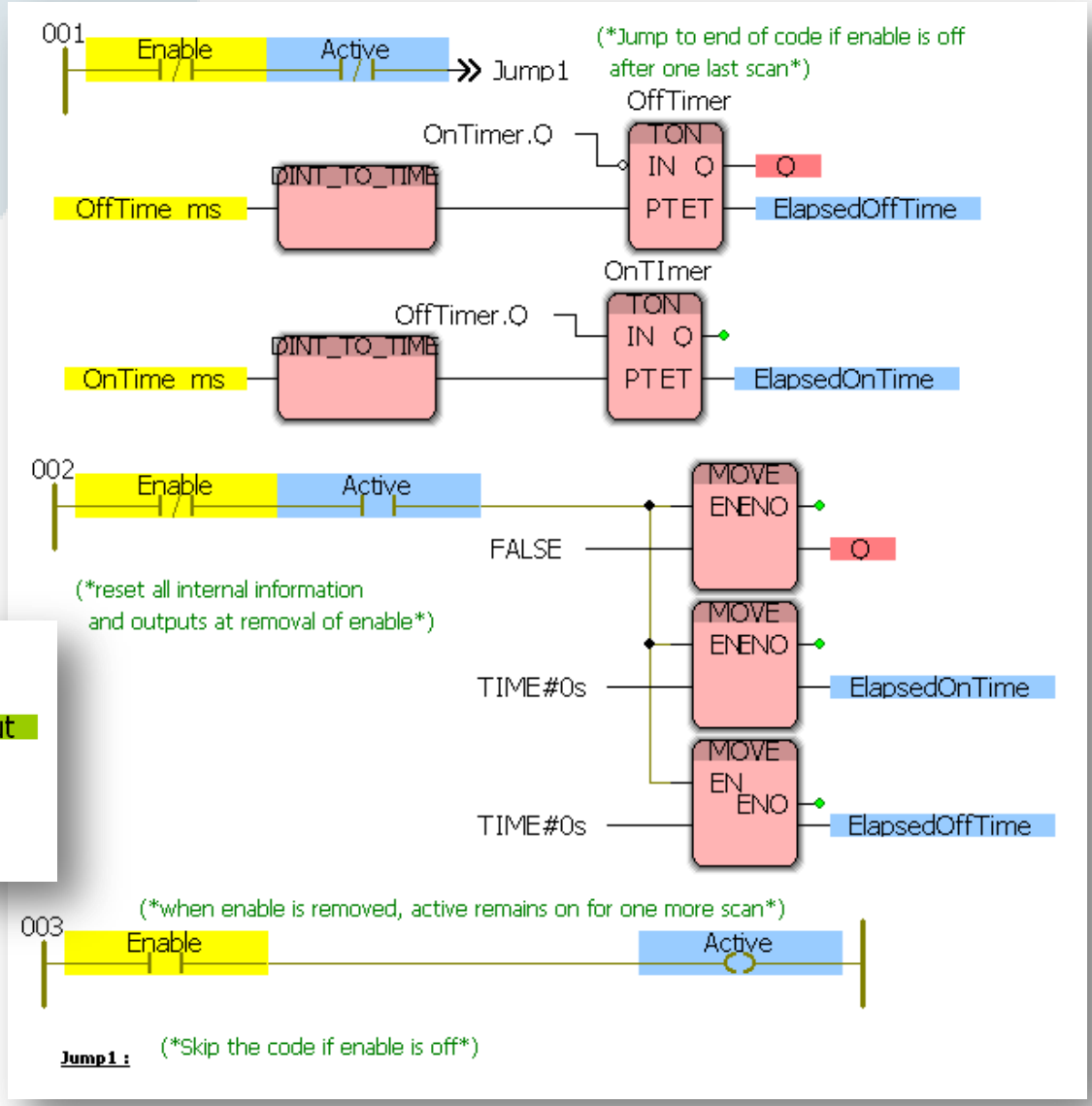
- *Insert YTTS_Blinker into the Main program POU*
  - *Cause Two outputs to blink independently*



Function Block with Enable

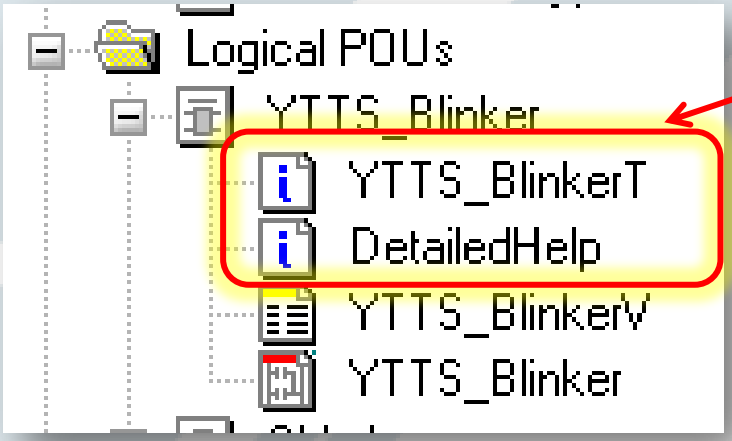
- *Jump to Label*
  - *For efficiency*
- *Active*
  - *To clear data after execute*



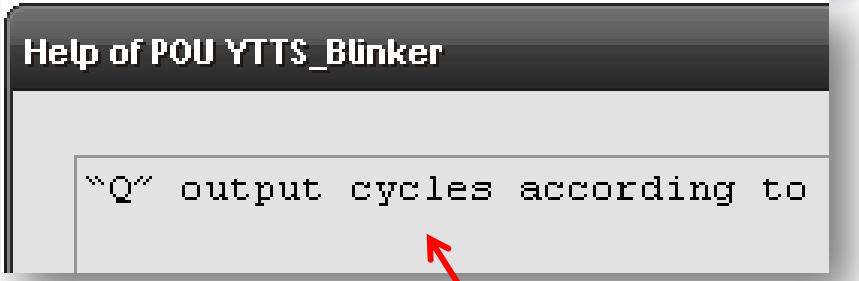


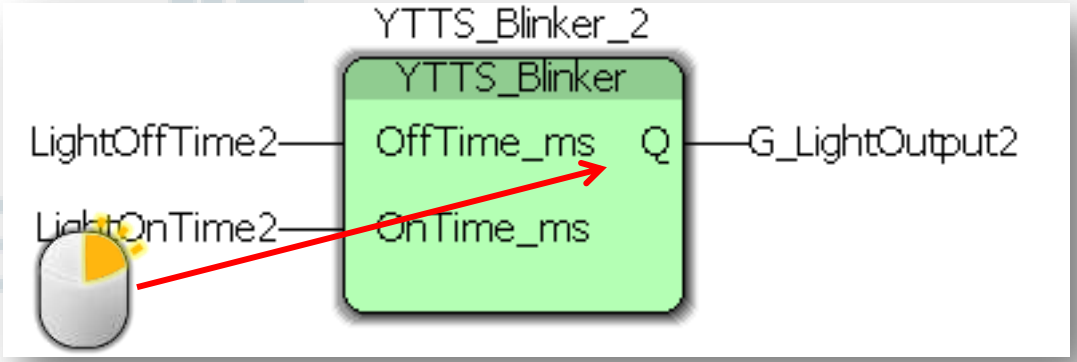
User Function Block Help

- *Option 1: Description Worksheet in Function Block Program*



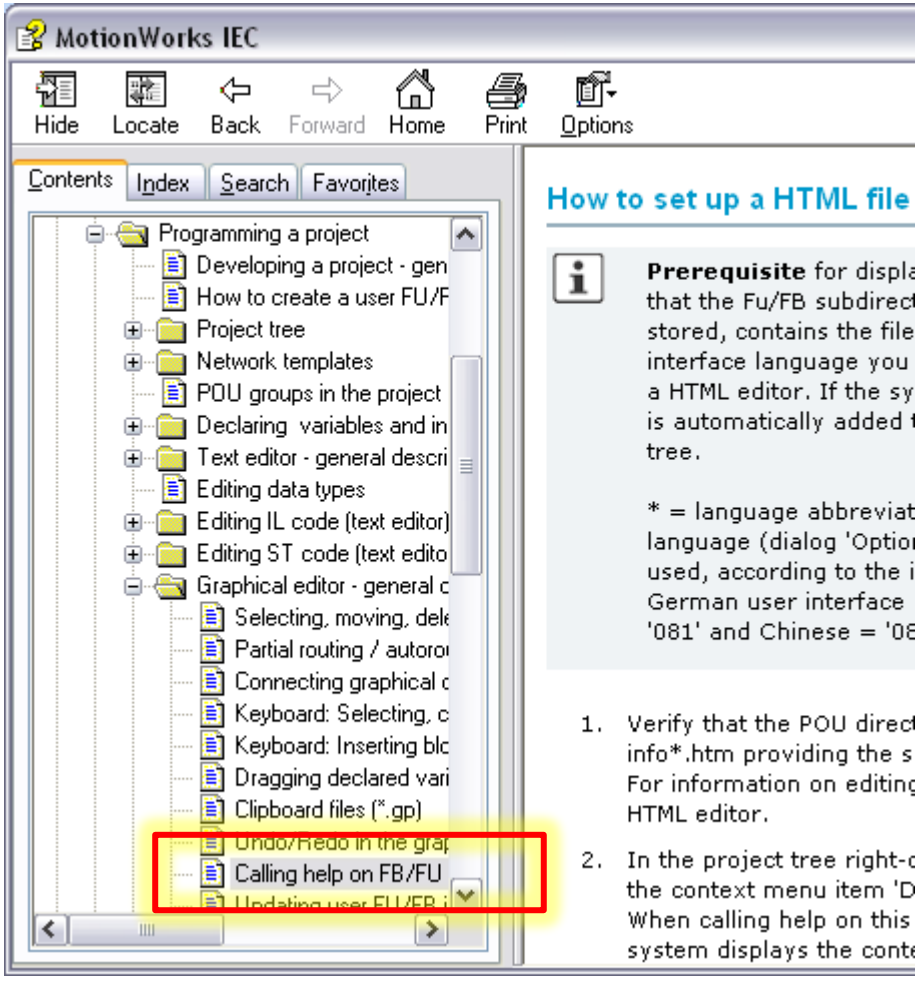
Help = Text from "Description Worksheet "





User Function Block Help

- *Option 2: *.HTML file*
  - *See Motionworks IEC Help*
  - *Search "HTML"*
  - *Graphics*
  - *Language Codes*



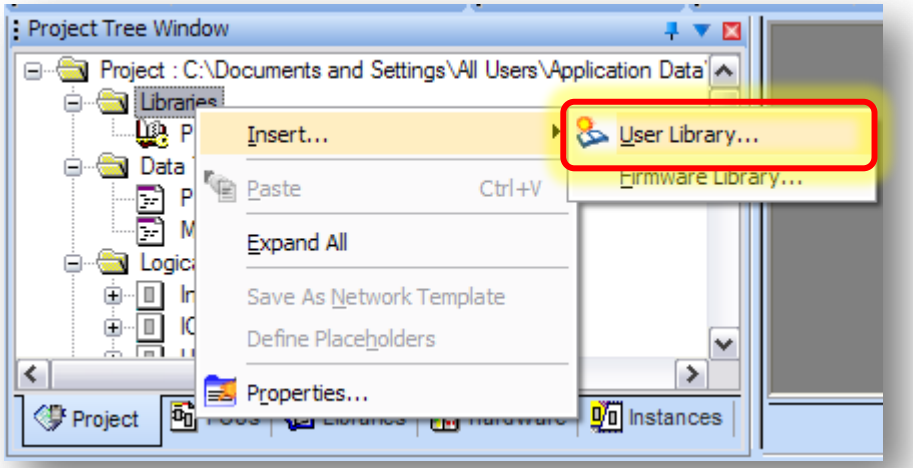
Driving Value

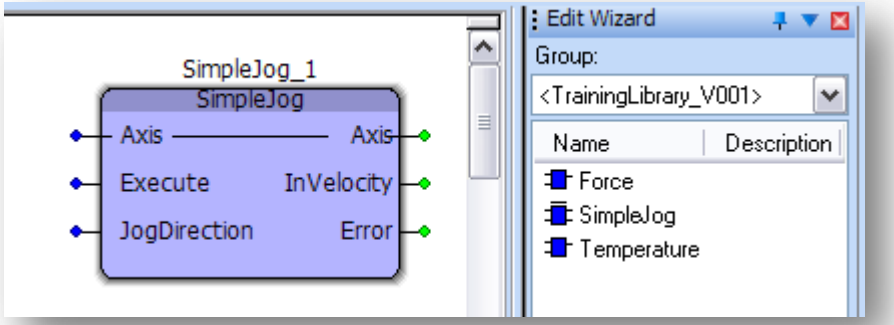
User Libraries

Process Overview
Create a Library
Use a Library
Zip / Unzip
Delete Datatypes

YASKAWA

- *Insert Another Project*
  - *Library = any project*
  - **.mwt (or *.mwe)*
- *Library Data Imported*
  - *User FU & FB POUs*
  - *Program POUs*
  - *Data Types*
  - *NOT global variables!*
- *Organization*
  - *Specific projects for library use*
  - *Revision number in project name*
  - *Prefix (ex: YTTS_)*





Prepare Library Project

- *Refer to Quick Reference Guide*
  - *Save project in "Libraries" folder under new name*
  - *Clean up and make the project*
  - *Save as ZIP for reference*

YASKAWA

Quick Reference Guide

MPiec Series Controllers

Contents

Startup Procedures
- Set the Front Panel Switches
- Open a saved project
- Start a new project
- Establish ethernet communication with controller (web interface)
- Go online with controller
- Download project to controller
- Set program to auto-start
- Reset an MP2300Siec system back to factory settings?
- Confirm motion is possible on each axis
- Tune the motors

Maintenance Procedures
- Update the firmware
- Determine/Set unknown IP address in controller
- Extract/Open the project from controller
- Replace the controller
- Replace the servopack
- Replace the servo motor

Analysis & Troubleshooting
- Monitor Program
- Watch Variables in a List (Monitor window)
- Use Logic Analyzer (AKA, graph, trace, scope)
- Diagnose Alarm Status

Modbus/TCP and EthernetIP
User Libraries
OPC Server

User Libraries

Overview

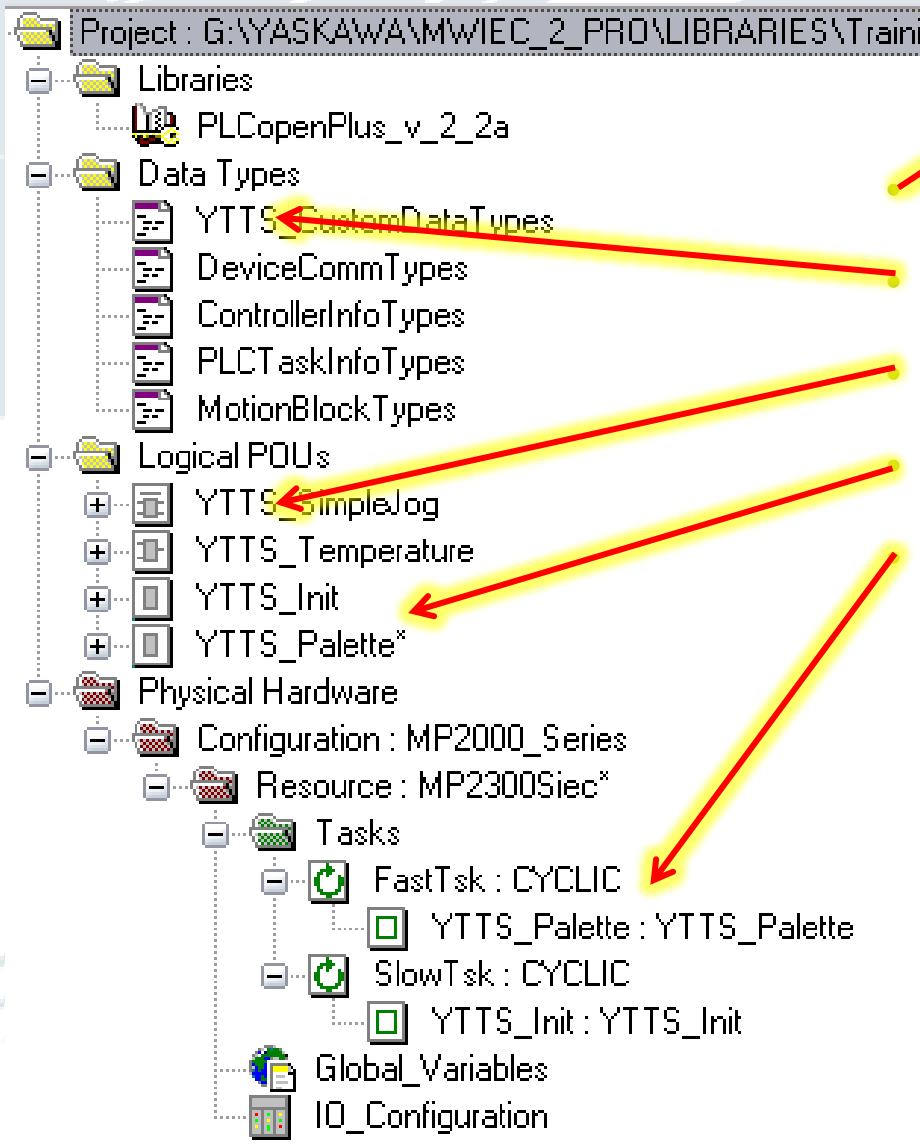
User Libraries are just normal project files imported into another project file as a library.
"Application Code Toolboxes" from Yaskawa are simply User Libraries created at Yaskawa

Create a Library

Step	Description	Detail
1	Rename the project and save in the Libraries folder (Optional, Best Practice)	File -> Save As.File -> Save Project As / Zip Project As... Navigate to Libraries folder Edit the file name to reflect the intended usage as a user library and revision control. For example, "TrainingLibrary_V001"
2	Delete unnecessary POUs, Tasks, Datatypes (Optional, Best Practice)	In project tree leave one LD program POU with an instance of each user FU/FB. Leave one ST program POU with data initialization. Leave one task to run both programs (for compile).
3	Prefix all POUs, Tasks, Datatypes (Optional, Best Practice)	Example Prefix: "YTTS_". Typical to leave LD programs YTTS_Palette and YTTS_Init running in YTTS_Tsk, using YTTS_CustomDatatypes.
4	MAKE the library project	Click "MAKE" and resolve all errors. A project that has only been renamed requires a new "MAKE"
5	Save the library project as ZIP (Optional, Best Practice)	File -> Save Project As / Zip Project As... Navigate to the Libraries folder (for organization purposes) Choose ZIP as the file type Best Practice: Do NOT change the name when zipping. Change the name in Step1 and re-MAKE first. Under "Zip Options" check boxes for "User Libraries" and "Front-end Code". Click the Zip button. *Single portable library file with revision name is produced*

The user library contains all POUs from the library project. Even program POUs can be reused. More common and useful is to only reuse FU and FB.

Prepare Library Project



Revision In Project Name

Data Type Prefix

POU Prefix

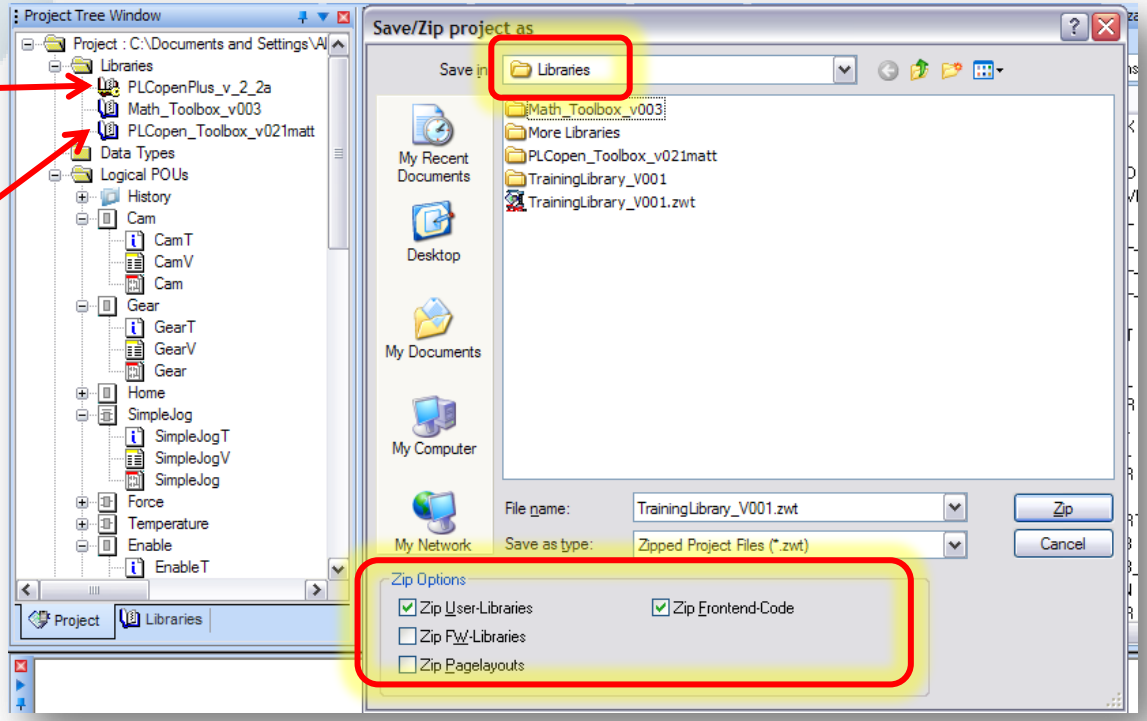
2 Program POUs

1-2 Tasks

- *Every task requires a program*

Zip Compiled Library

- *Zip to Libraries Folder*
  - *Keep Organized*
  - *Single File is portable*
- *FW-Libraries (red)*
  - *Ex: PLCopenPlus*
  - *Part of MWiec*
- *User Libraries (Blue)*
  - *Zip these libraries in this project*
- *Frontend-Code*
  - *Zip the result of "Make"*
- *Pagelayouts*
  - *Not used by Yaskawa*



Default check "Zip User-Libraries" and "Zip Frontend-Code"

Add Library

- *New Project*
  - *Insert user library*
  - *Add user function and function block to a POU*

Refer to the Quick Reference Guide

YASKAWA

Quick Reference Guide

MPiec Series Controllers

Contents

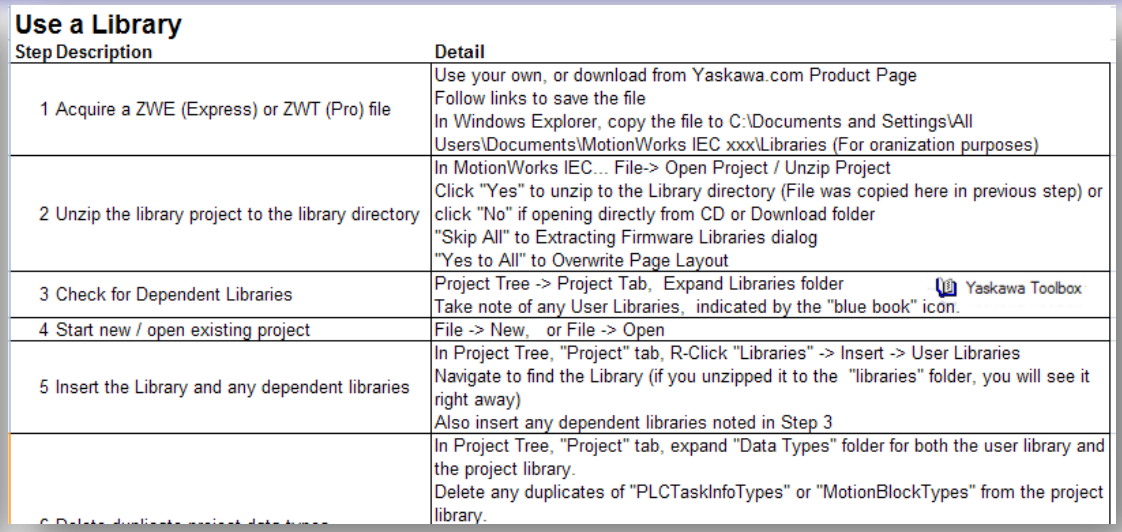
Startup Procedures
- Set the Front Panel Switches
- Open a saved project
- Start a new project
- Establish ethernet communication with controller (web interface)
- Go online with controller
- Download project to controller
- Set program to auto-start
- Reset an MP2300Siec system back to factory settings?
- Confirm motion is possible on each axis
- Tune the motors

Maintenance Procedures
- Update the firmware
- Determine/Set unknown IP address in controller
- Extract/Open the project from controller
- Replace the controller
- Replace the servopack
- Replace the servo motor

Analysis & Troubleshooting
- Monitor Program
- Watch Variables in a List (Monitor window)
- Use Logic Analyzer (AKA: graph, trace, scope)
- Diagnose Alarm Status

Modbus/TCP and EthernetIP
User Libraries
OPC Server

MotionWorks IEC



Use a Library

Step Description	Detail
1 Acquire a ZWE (Express) or ZWT (Pro) file	Use your own, or download from Yaskawa.com Product Page Follow links to save the file In Windows Explorer, copy the file to C:\Documents and Settings\All Users\Documents\MotionWorks IEC xxx\Libraries (For oranization purposes)
2 Unzip the library project to the library directory	In MotionWorks IEC... File-> Open Project / Unzip Project Click "Yes" to unzip to the Library directory (File was copied here in previous step) or click "No" if opening directly from CD or Download folder "Skip All" to Extracting Firmware Libraries dialog "Yes to All" to Overwrite Page Layout
3 Check for Dependent Libraries	Project Tree -> Project Tab, Expand Libraries folder Yaskawa Toolbox Take note of any User Libraries, indicated by the "blue book" icon.
4 Start new / open existing project	File -> New, or File -> Open
5 Insert the Library and any dependent libraries	In Project Tree, "Project" tab, R-Click "Libraries" -> Insert -> User Libraries Navigate to find the Library (if you unzipped it to the "libraries" folder, you will see it right away) Also insert any dependent libraries noted in Step 3
	In Project Tree, "Project" tab, expand "Data Types" folder for both the user library and the project library. Delete any duplicates of "PLCTaskInfoTypes" or "MotionBlockTypes" from the project library.

The user library contains all POUs from the library project. Even program POUs can be reused. More common and useful is to only reuse FU and FB.

ENGINEERING EXPERTISE

EASY TO
WORK WITH YASKAWA™

INOLOGICAL
INNOVATION

QUALITY
PRODUCT